NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A GRAPHICAL USER INTERFACE FOR THE LOW COST COMBAT DIRECTION SYSTEM

by

Michael Grey Stockwell

September 1991

Thesis Advisor: Dr. Valdis Berzins

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Computer Science Dept. Naval Postgraduate School; 6b. OFFICE SYMBOL (if applicable): CS	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
A GRAPHICAL USER INTERFACE FOR THE LOW COST COMBAT DIRECTION SYSTEM

12. PERSONAL AUTHOR(S)
Stockwell, Michael Grey

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM 09/90 TO 09/91	1991, September, 16	264

16. SUPPLEMENTARY NOTATION: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Combat Direction Center, Software Engineering, User Interface, Graphics

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

A technology vacuum exists in the small combatant and support ship units of the U.S. Navy. Newer combatant vessels are typically outfitted with state of the art Combat Direction and information processing systems, while their older sisters in the fleet are still using manual methods to do the same tasks.

These shipboard tasks, which include contact management, moving geometry calculations, intelligence compilation, area plotting and navigation, are all performed manually on-board older fleet units. These manual methods are extremely slow, are subject to potentially disastrous errors and require intensive training to perform correctly. The current level of computer technology allows the automation of these tasks, providing an instant solution to the user, with a far smaller error rate than could be expected of manual methods.

The prime objective of this thesis, The Low Cost Combat Direction System (LCCDS) User Interface, is to define the functional goals, constraints and general system design of a graphical user interface suitable for use aboard U.S. Navy vessels. Particular emphasis is placed on the design and implementation of a graphical display system, in the Ada programming language, necessary to interface the user to the capabilities of the LCCDS as a whole.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL: Valdis Berzins	22b. TELEPHONE (Include Area Code): (408) 646-2461	22c. OFFICE SYMBOL: CS/Be

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Approved for public release; distribution is unlimited

A GRAPHICAL USER INTERFACE FOR THE LOW COST COMBAT DIRECTION SYSTEM

by

Michael Grey Stockwell
Lieutenant, USN
B.S., Chapman College, 1983

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 1991

Author: _____
Michael Grey Stockwell

Approved By: _____
Valdis Berzins , Thesis Advisor

_____
Luqi, Second Reader

_____
Robert B. McGhee, Chairman,
Department of Computer Science

ABSTRACT

A technology vacuum exists in the small combatant and support ship units of the U.S. Navy. Newer combatant vessels are typically outfitted with state of the art Combat Direction and information processing systems, while their older sisters in the fleet are still using manual methods to do the same tasks.

These shipboard tasks, which include contact management, moving geometry calculations, intelligence compilation, area plotting and navigation, are all performed manually on-board older fleet units. These manual methods are extremely slow, are subject to potentially disastrous errors and require intensive training to perform correctly. The current level of computer technology allows the automation of these tasks, providing an instant solution to the user, with a far smaller error rate than could be expected of manual methods.

The prime objective of this thesis, The Low Cost Combat Direction System (LCCDS) User Interface, is to define the functional goals, constraints and general system design of a graphical user interface suitable for use aboard U.S. Navy vessels. Particular emphasis is placed on the design and implementation of a graphical display system, in the Ada programming language, necessary to interface the user to the capabilities of the LCCDS as a whole.

TABLE OF CONTENTS

I. INTRODUCTION

This thesis addresses the design and implementation of a user interface, in Ada, for a near-real-time tactical information system. It is a part of ongoing research for the development of a Low Cost Combat Direction System (LCCDS) at the Naval Postgraduate School.

A. BACKGROUND

1. Low Cost Combat Direction System

Development of the LCCDS is sponsored by the Naval Sea Systems Command (NAVSEA) as an economical alternative for providing Combat Direction System (CDS) capabilities to non-combatant ships. LCCDS requires a platform independent system supporting graphical windowing capabilities implemented on commercially available workstations. The ultimate goal of the system is to provide enhanced information support resources to commanders of naval non-combatant ships [SOW88].

2. Ada Programming Language

The United States Department of Defense (DOD), recognizing the need for a single high level language for its sponsored applications, has mandated the use of Ada for all Defense related projects. Although Ada is an admirable language for embedded systems, it has no direct capability to provide windowing or the graphical support necessary to implement an effective user interface.

3. X11 Windowing System

The X11 windowing system, developed by the Massachusetts Institute of Technology (MIT), is the most widely supported graphical window system meeting the platform independence requirements of the LCCDS project. The X11 system, currently on revision 4, provides a very versatile environment for a programmer to implement an effective user interface to any application. As X11 is written in the C programming language, there are difficulties when trying to merge an Ada application which calls the C functions necessary to provide the graphics involved in a typical user interface.

4. Software Technology For Adaptable, Reliable Systems

The Software Technology for Adaptable, Reliable Systems (STARS) program is a US Air Force Systems Command sponsored project, developed by Unisys Defense Systems, which provides the necessary interface between an Ada application and the C language functions into the X11 Protocol. The Ada/Xlib bindings developed under the STARS program provide an Ada data structure allowing the binding of Ada subroutines to their C equivalents. The bindings provide a complete mapping to all the X11R4 window primitives.

5. Transportable Applications Environment Plus

The Transportable Applications Environment Plus (TAE+) is a National Aeronautics and Space Admistration (NASA) sponsored project, developed by Century Computing , Inc. TAE+ is a complete graphics application environment, providing a suite of tools and programming aids allowing the developer to define a graphics

application directly on the display screen. After completion of an application design, TAE+ will automatically generate either C or Ada compilable code , which can either stand alone as a running application or may be subdivided into packages for inclusion into another program.

6. Graphical User Interface

A user interface for a CDS environment must present the user with a wide variety of information for each display item encompassing elements of position, identification, classification and the attributes of course, speed etc. The wide variety of these items make a text based information display too difficult for the user to comprehend at a glance. A graphical interface can represent multiple items of information within a single symbol, presenting the user with a comprehensive display of information, easily understood with even a brief glance at the display screen.

B. STATEMENT OF THE PROBLEM

A graphical interface to the LCCDS is required to present the system user with a symbolic representation of information, easily understood with even a brief glance at the display. The functions of data input and retrieval must be simple and easy enough to be understood without resorting to technical manuals or help displays. Finally, but certainly not the most minor consideration, is the speed of system response. The interface must be able to respond immediately to user requests, changes in the system environment and

new information entering the system. This near-real-time response rate places significant timing constraints on the interface that cannot be under-emphasized.

This thesis implements a design of a real time system, using off the shelf hardware and readily available Ada software resources, for a typical CDS. The methodology used in this project, and the implementation structure employed to present the user interface can be readily transferred to any project requiring a graphical user interface requiring rapid response times.

C. SUMMARY OF PREVIOUS WORK

1. Seveney/Steinberg 1990

A joint project describing the display and timing requirements for the LCCDS. Full details are available in [SS90].

2. Sun 1990

Research into the various tools utilizing the X11 Protocol, with emphasis on C language toolkits. Full details are available in [SUN90].

3. Coskun/Kesoglu 1990

A design and implementation of a text based C3I system utilizing the TAE+ application generator. Full details are available in [CK90].

4. Bolick/Irwin 1991

A design and implementation of a Tactical Database for the LCCDS, researched in parallel with this project. Full details are described in [BI91].

D. RESEARCH APPROACH

1. Requirements Study

This research began with a comprehensive study regarding the elements of information required to present a CDS user with all the support necessary for an effective system. Extensive conversations with prospective users and surveys of Naval Officers with non-combatant ship experience provided a detailed list of items desirable in a typical CDS for shipboard use. A summary of those items is given in Chapter II.

2. Design Of The Information Display

Using the list of items desirable in a CDS, a design to implement all functions required for the user to access the necessary information was devised. Those items determined to be of immediate importance were assigned display screens and those screens kept up to date without any user interaction. The items of secondary importance were assigned to menu options allowing the user to call those functions as required. A summary of the methodology and the final design decisions made is described in Chapter III.

3. Survey Of Available Implementation Tools

After completion of the initial design, a survey of the tools available to implement that design was made. Only a very narrow range of tools currently exist to port Ada applications into the X11 Protocol. A summary of those tools is given in Chapter IV.

4. Implementation And System Integration

Implementation of the final interface design was made utilizing the STARS Ada/Xlib bindings and the NASA TAE+ application generator. The interface was then linked with the Bolick/Irwin LCCDS Tactical Database described in section C. Timing and response tests were performed with the results described in Chapter IV.

II. INTERFACE REQUIREMENTS

An effective interface reflects the needs of the end users. To ensure that the initial design specification detailed in the Seveney/Steinberg thesis [SS90] accurately depicts the needs of the Naval Surface Forces, a survey was taken of 35 Surface Warfare Naval Officers. Fifteen of these officers had prior experience in surface non-combatant units. As a result of this survey, a number of areas not previously mentioned in Seveney/Steinberg were raised. The issues raised are presented in summary form, grouped by function.

A. PLOTTING FUNCTIONS

1. Station, Screen And Formation Plotting

It is often necessary for a ship underway, particularly when operating with other ships, to be positioned in one particular area of the ocean. This area is often moving along with another unit and is called a station. The ability to plot the boundaries of these stations, allowing the user to see at a glance if the ship is "on station" was highly recommended as an addition to the interface.

2. Operating Area And Grid Plotting

When operating in normal fashion, ships are often assigned large areas of ocean to conduct their operations. There are several different predefined areas, known as Operating Areas or Grids, a ship may be assigned to. Ships are typically tasked to be in a

particular operating area, or assigned to meet another unit in another area. The ability to plot these areas was desired by the users as a convenience to determine the necessary course and speed requirements to transit in and out of these areas.

3. Amphibious Warfare Area Plotting

It is a primary task of non-combatant ships to provide support to combatant units engaged in Amphibious Assault. Elaborate diagrams, detailing the proper position of all units required for the operation are currently drawn up by hand and must be referred to often. The ability to plot the diagram on the display screen was highly encouraged.

4. Misc. Plotting Functions

Several other plotting type functions were recommended, including PIM tracks, great circle voyage planning, lines of demarcation, anchoring approach diagrams, aircraft corridors, commercial air lanes and nuclear attack avoidance zones. Most of the items discussed were met with wide acceptance with the majority of the user group.

B. DRAWING FUNCTIONS

1. User Defined Line Drawing Functions

It is often necessary for a ships officer to draw lines to delineate areas or to simply provide a frame of reference to conduct an operation. The capability to draw lines on the display screen, either anchored to geographical locations or to other moving objects was stressed as vitally important.

C. RECOMMENDATION FUNCTIONS

1. Contact Avoidance Or Intercept Recommendations

The ability to have an immediate course and speed recommendation to either avoid another object or to intercept that object was deemed as extremely helpful to the user.

2. Air Operations Recommendations

Flight operations from a surface ship generally require a minimum relative wind speed and direction in relation to ownship to ensure the safety of the flight crew and platform. The ability to generate a course and speed recommendation to achieve the necessary wind conditions was mentioned as being very advantageous.

3. Time And Distance Recommendations

It is always the case that a surface ship is tasked with being at a particular spot at a particular time. The ability to get a recommended course and speed to reach a certain location at a given time was stressed as being of great importance.

D. INTELLIGENCE FUNCTIONS

1. Object Offensive And Defensive Capabilities

There is an information gap, particularly among junior officers, as to the capabilities of the warships of other nations. The ability to call up the characteristics of

any identified object from a central database was unanimously agreed upon as a desirable attribute of the LCCDS.

E. LISTING AND TEXT ENTRY / RECALL FUNCTIONS

1. Routine And Emergency Checklists

Every ship in the U.S. Navy has certain procedures defined to respond to both routine operations and basic emergencies. These procedures are generally kept in written form as a series of checklists. The ability to maintain these checklists on line and record completion of each list item was regarded highly by the majority of the group.

2. Watchbills And Daily Routine Summaries

The ability to maintain on line the ship's daily schedule and the watchbills in effect for the various shipboard departments was desired by the users as a convenient reference to the Officer of the Deck aboard any surface ship.

F. ENCODING AND DECODING FUNCTIONS

1. Allied Tactical Publication Signal Encoding / Decoding

Any time ships are communicating with each other over non-secure voice radio, particularly concerning movement orders or intentions, a set of signals defined by the Allied Tactical Publication 1 (ATP-1) are employed to minimize the chance of a signal being misunderstood. These signals are cryptic and must be encoded/decoded

using manual procedures at every occurrence. Although not thought of by the group as being necessary, it was generally agreed upon as a "nice to have" feature for the LCCDS.

III. USER INTERFACE DESIGN

Since the requirements for this project specified a windowed design, the design was tailored to fit the maximum number of functional requirements into a set of windowed displays. The process began by analyzing all requirements for two items: user interaction and information display.

The requirements needing user interaction were separated into one logical grouping and those requiring information display into another. It became immediately apparent that most requirements have elements of both interaction and display as the two groupings were virtually identical.

The effort, however, was not wasted. The two logical groupings became the basis for later decisions regarding menus and displays. The requirements needing user interaction led to menu options and those needing information display were assigned a display window.

Due to the close relationship between the required user interaction and the information display, categorizing the design process as a series of phases is difficult. The method used to complete the design can best be described as having four parts, with each of the parts requiring frequent excursions into the others.

- Functional requirements classification
- Functional category analysis
- Information display analysis
- Window sizing and position analysis

The figures shown in this chapter reflect the final design decisions made for the implementation of the various menu and information display items.

A. FUNCTIONAL REQUIREMENTS CLASSIFICATION

The design phase of this project was begun by analyzing the initial requirements described in Seveney/Steinberg [SS90] and the additional functional requirements described in Chapter II. A list of overall requirements, with an attendant list of operations required to implement them, was developed and classified into logical, top level categories combining function, operations and required user interaction. A total of nine categories emerged from this phase.

- Track functions
- Plotting functions
- User alert parameters
- Display filtering parameters
- Intelligence input and display functions
- List input and retrieval functions
- Coding/decoding functions
- Range scaling functions
- System input and display default parameters
- Navigation functions

Each of these categories contain elements of both user interaction and information display. To handle the required user interaction, one atomic label, to be used as a menu title, was drawn from each category.

At this point, the number of menu items forced a decision regarding menu presentation. A single menu bar, as shown in Figure 3-1, with 9 pulldown type items was decided upon as the best presentation for the end user.

Figure 3-1. The Menu Bar

One additional menu item, allowing the capability to exit cleanly from the Information System, was placed into the title window shown in Figure 3-2.

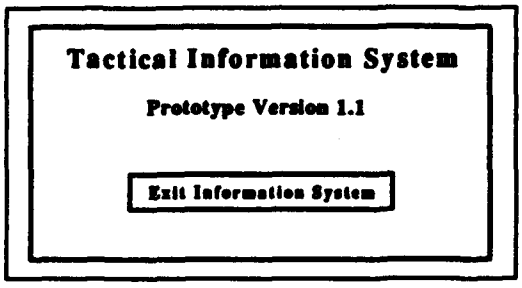

Figure 3-2. TIS Title Window

B. FUNCTIONAL CATEGORY ANALYSIS

Each of the atomic menu options was analyzed to determine the minimal set of operations required to fulfill all elements of the requirements as classified in section A. Each set of operations within a menu option was limited to a size of five items to minimize menu congestion.

1. Track Functions Analysis

This category covers all requirements dealing with the Track type objects. There are only three primary operations a user must be able to perform on objects of this type to meet all the requirement objectives.

- Add a new track object
- Delete an existing track object
- Modify the attributes of an existing track object

These three operations became the options for the Menu Bar pulldown button ***Track*** as shown in Figure 3-3.

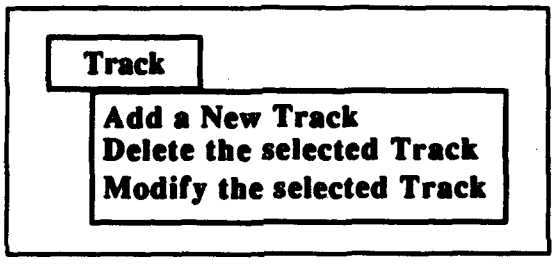

Figure 3-4. Track Pulldown Options

a. Add A New Track

In order to define a new track, all the attributes of a track object must be input by the user. The attributes defined for a track in the prototype system are:

- Position
- Course
- Speed
- Category (Track, Point)
- Classification (Hostile, Friendly, Neutral, Unknown)
- Type (Surface, Air, Sub-Surface)
- Height (or Depth)
- Identification (Country, Platform type, Name)
- Misc. comments

(1) Position. Position can be input in 3 different ways.

- Entering the position in Latitude/Longitude figures
- Entering the position as a Bearing and range from Ownship
- Entering the position by selecting a point on the display screen

To handle the three possible cases, a separate menu window was constructed with radio buttons denoting the three mutually exclusive cases as shown in Figure 3-4.

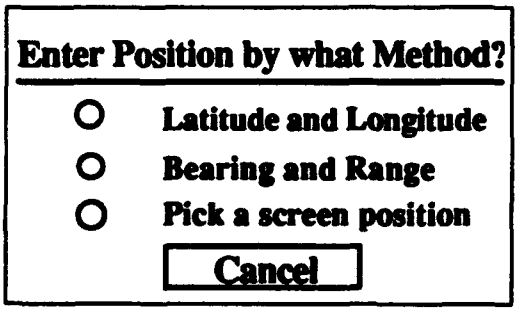

Figure 3-4. Enter Position Method

Figures 3-5 and 3-6 handle the first two position entering possibilities and Figure 3-7 appears when the third is selected.

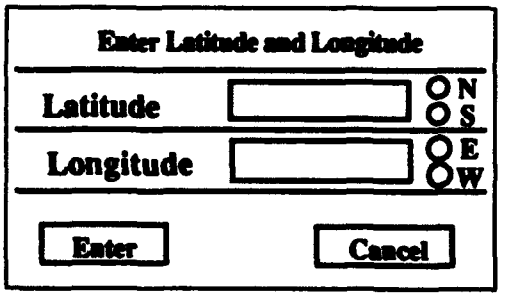

Figure 3-5. Enter Lat/Long Position

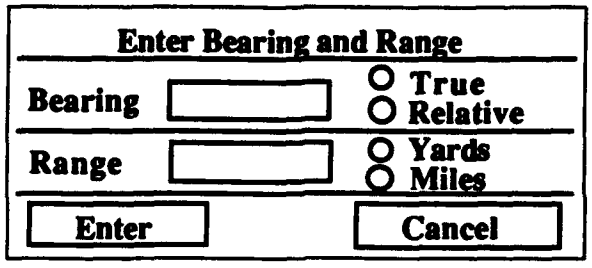

Figure 3-6. Enter Bearing/Range Position

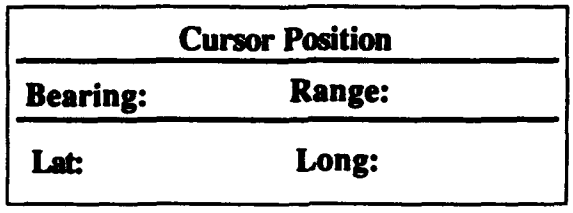

Figure 3-7. Cursor Tracking Window

The Cursor Tracking window will return the virtual bearing and range of the cursor from Ownship and the position of the cursor in Latitude and Longitude coordinates, disappearing when a screen position is selected.

After the positioning method is selected, and the position input, a series of additional menu windows are presented to the user to input and/or select the remaining attributes. Figures 3-8 through 3-12 show the remaining sequence of menu screens.

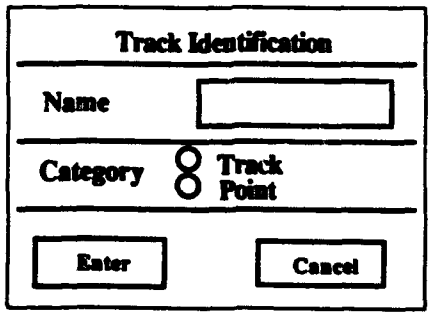

Figure 3-8. Name Attribute

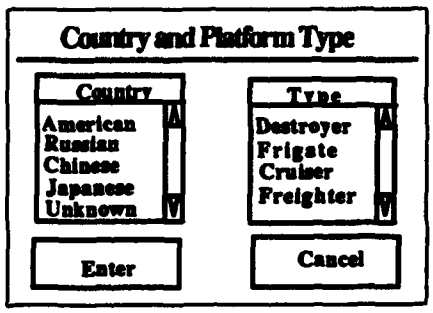

Figure 3-9. Country and Platform Attributes

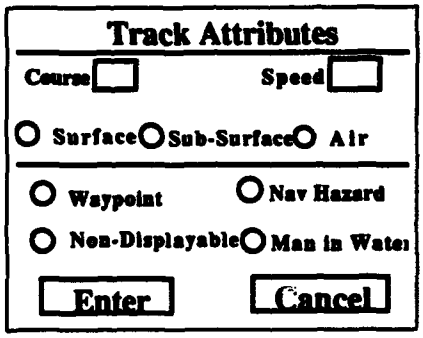

Figure 3-10. Misc Attributes

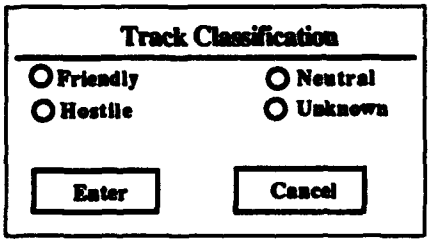

Figure 3-11. Classification Attribute

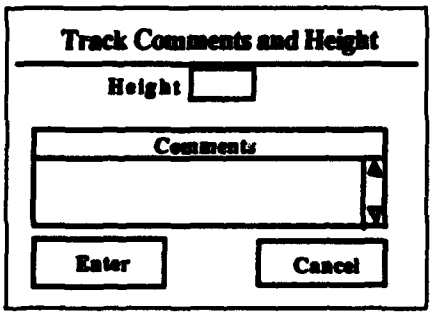

Figure 3-12. Comments and Height Attributes

b. Delete The Selected Track

When this option is selected, the system will delete the selected, (or "hooked") track. In the event that no track is selected, a prompt will appear to instruct the user to pick the track to be deleted. A window will appear containing the information on the selected track and asking the user to confirm the operation. Figures 3-13 and 3-14 illustrate the progression of windows presented to the user for this operation.

Figure 3-13. Select Warning

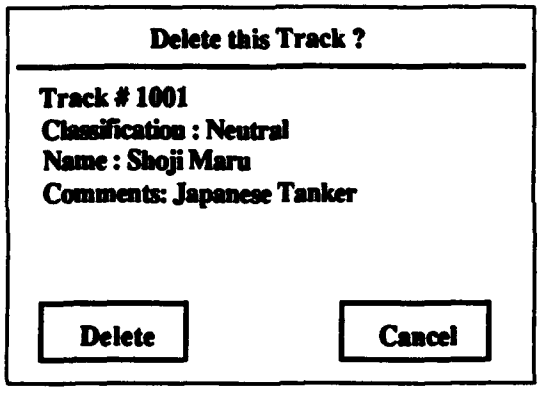

Figure 3-14. Delete Confirm

c. Modify The Selected Track

This option will allow the user to modify any parameter of the selected Track. If no Track is "hooked" the user will be prompted to do so as in Figure 3-13. A menu window, as shown in Figure 3-15, containing checkboxes will appear asking the user to select the parameters to be changed. The user will choose the different parameter categories to be changed and upon selecting the ***Enter*** button, a selective progression of

data input windows from Figures 3-4 through 3-12 will be presented to reflect the choices the user has made.

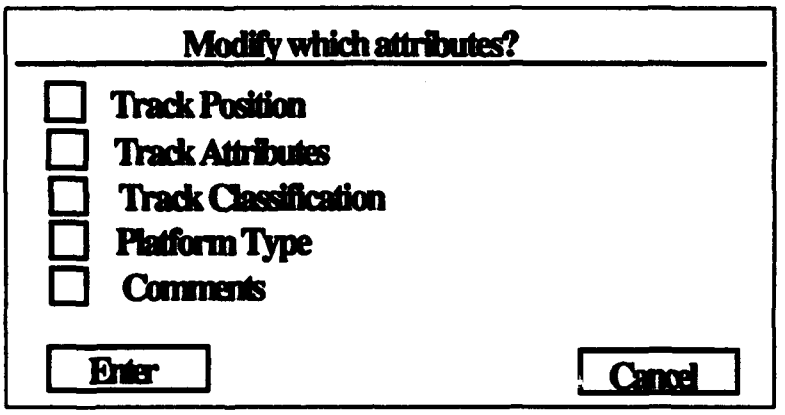

Figure 3-15. Modify Select

2. Plotting Functions Analysis

The myriad of different plotting functions defined in the requirements called for a different design approach. All the plotting requirements were studied and classified into three logical categories.

- Maps and Grids
- Zones and Areas
- Sectors and Formations

These three categories became the options for the Menu Bar pulldown button Plots as shown in Figure 3-16.

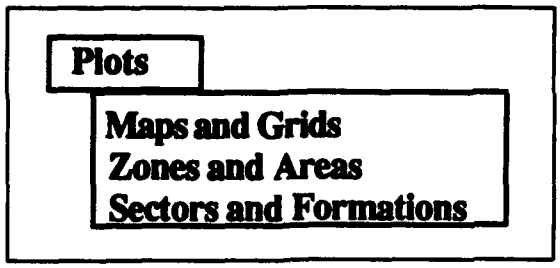

Figure 3-16. Plots Pulldown Options

a. Maps And Grids

There are two requirements specified for this area, one for displaying shore line maps and one for displaying UTM grid lines. Of course, if the ability exists to activate the maps and grids, there must be the corresponding capability to deactivate them. The four operations defined for this option are:

- Activate Shoreline Mappings
- Deactivate Shoreline Mappings
- Activate UTM Grid Lines
- Deactivate UTM Grid Lines

The four operations translated to 2 corresponding check box options as shown in Figure 3-17. As the shoreline mapping and grid displays are not supported in the prototype, no functionality was attached to these options but was included for future development.

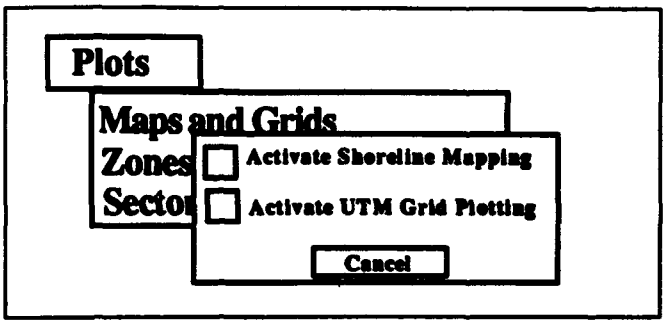

Figure 3-17. Maps and Grids Options

b. Zones And Areas

There are three kinds of zones and two types of areas defined in the requirements. Two of the zone types were compressed to one entry as the plotting mechanics are identical. The operations defined for this category are:

- Plot a Missile Engagement / Fighter Engagement Zone
- Plot a "Keep out" Zone
- Plot an Operating Area
- Plot an Amphibious Assault Area Diagram

The four operations were displayed as radio button choices in a sub-menu to the Zones and Areas pulldown option as shown in Figure 3-18. No further functionality was defined for these operations as they are not supported in the prototype.

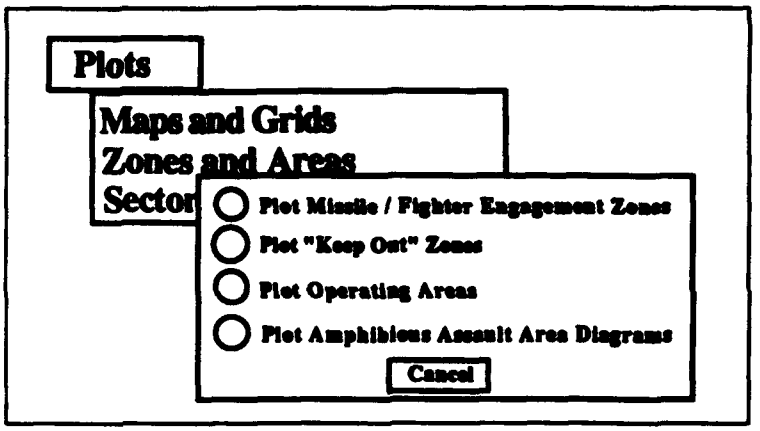

Figure 3-18. Zones and Areas Options

c. Sectors And Formations

There are two requirements specified for this category, one for sectors and one for formations. The operations required are:

- Sector Screen Plotting
- Formation Diagram Plotting

The two operations were included in a sub-menu to the Sectors and Formations pulldown button as shown in Figure 3-19. No further functionality was designed as the operations are not supported in the prototype.

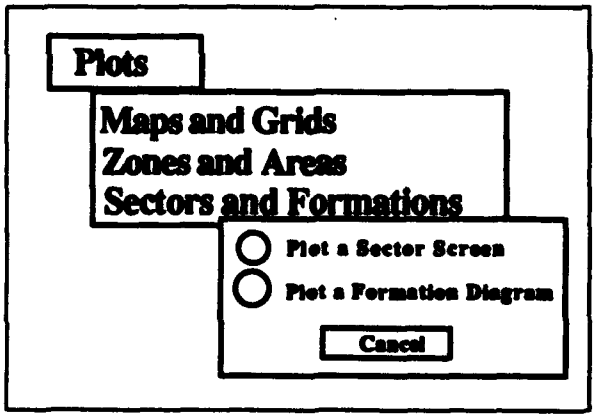

Figure 3-19. Sectors and Formations Options

3. Alert Functions Analysis

All active alerts are defined in the System Setup menu so the only operations required for this category are the following:

- Visual Alerts only
- Audio and Visual Alerts
- Disable all Alerts

The three operations became the pulldown menu options for the button ***Alerts*** as shown in Figure 3-20.

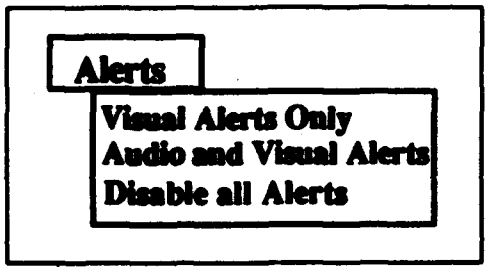

Figure 3-20. Alerts Pulldown Options

a. Visual Alerts Only

This option will disable the audio portion of the Alert functions. The only indication of an alert will be the text in the ***Alerts Display*** window. The Alert status will also be passed to the ***Screen Display*** and will result in the message "***Alerts : Visual***" being displayed on the bottom left side of the display.

b. Audio And Visual Alerts

This is the system default status and will result in both audio and visual cues when an ***Alert*** occurs. The ***Screen Display*** will contain the message "***Alerts : On***" at the bottom left side of the display.

c. Disable All Alerts

This option will disable both the audio cues and the input to the ***Alerts Display*** window. No alerts will be generated. The message "***Alerts : Disabled***" will appear at the bottom left side of the ***Screen Display***.

4. Display Filtering Functions

Like the ***Alert*** parameters, the ***Display Filter*** parameters are selected in the ***Set System Defaults*** menu section. Recognizing that the user may want to temporarily override the predefined filter values, two operations have been provided.

- Disable all Display Filters
- Activate Default Display Filters

The two operations became the options for the pulldown button ***Filters*** as shown in Figure 3-21.

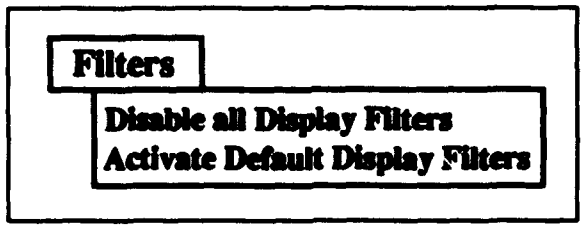

Figure 3-21. Filters Pulldown Options

a. Disable All Display Filters

This option will result in the overriding of all ***Display Filters*** selected in the ***Set System Defaults*** menu section. All objects arriving into the Integration System Database will be processed and displayed. The message "***Display Filters: Disabled***" will appear at the top of the ***Screen Display*** to remind the user of the Filter status.

b. Activate Default Display Filters

This option will restore the system ***Display Filters*** to the default condition as prescribed in the ***Set System Defaults*** menu section. The message "***Display Filters : Default***" will appear at the top of the ***Screen Display***.

5. Intelligence Functions Analysis

The Intelligence system works via a system of keyword entries into an Intelligence database. The only operations required to support this system are mechanisms to specify which of the Track attributes to use as the keyword entries to the database, and the ability to edit the database currently in use. The operations defined for this category are:

- Use Name for Intel Search
- Use Comments for Intel Search
- Use Country and Platform Type for Intel Search
- Edit Intel Database

The four operations formed the options for the Intel pulldown button as shown in Figure 3-22. No further functionality was included in this area as the prototype does not support Intelligence functions.

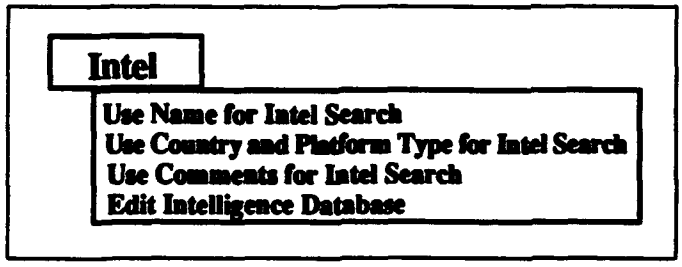

Figure 3-22. Intel Pulldown Options

a. Use Name For Intel Search

Name is the default Intel search parameter. The ***Track*** attribute "***Name***" will be passed to the Intel search routine each time a Track is selected from the Screen Display. The results of the search will be displayed in the ***Intelligence Display*** window. The message "***Intel : Name***" will appear in the lower right corner of the ***Screen Display*** to remind the user of the Intel search criterion.

b. Use Country And Platform For Intel Search

With this option selected, the ***Track*** attributes "***Country***" and "***Platform Type***" will be passed to the Intel search routine each time a Track is selected. The results of the search will be displayed in the ***Intelligence Display*** window. The message "***Intel : Platform***" will appear in the lower right corner of the ***Screen Display***.

c. Use Comments For Intel Search

This option will pass the Track attribute "***Comments***" to the Intel Search routine. Each word in the ***Comments*** string will be passed individually and scanned separately. Due to the comparative slowness of this operation, it will only act upon the

Track selected at the time this option is selected and the search operation will be forced to terminate at the next mouse / trackball button event. The results of the search will be displayed in the ***Intelligence Display*** window. The message "***Intel : Searching***" will be displayed in the Screen Display. At the termination of the search, the Intel message in the ***Screen Display*** will revert to the last default selected.

d. Edit The Intel Database

This option will allow the user to edit the Intelligence database. The location of the editing screen and the mechanics of the data input will be determined by the database implementor.

6. List Input And Retrieval Functions Analysis

This category covers all the requirements dealing with display and editing of text objects, such as Ship's Bills, Checklists, Watchbills and the like. There are two operations required for this category.

- View a List
- Edit a List

The two operations became the entries for the ***Lists*** pulldown menu as shown in Figure 3-23.

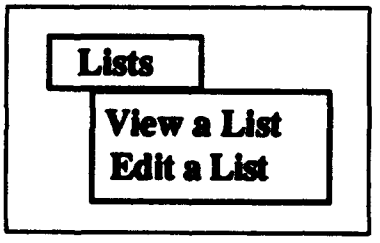

Figure 3-23. Lists Pulldown Options

a. View A List

This option will present the user with a list of radio button options containing the names of all the text objects currently in the system. Selecting one of the options will cause a text window, containing the desired text, to appear on the right side of the screen, overlaying the various information display windows. Figures 3-24 and 3-25 show the text select and display windows respectively.

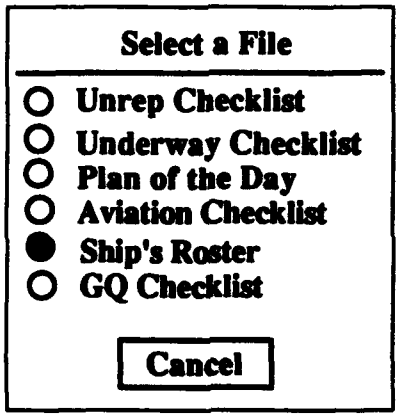

Figure 3-24. File Select Window

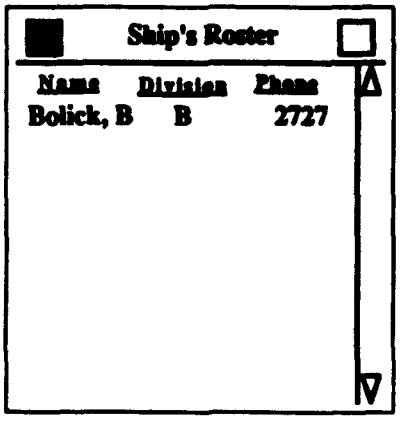

Figure 3-25. Text Display Window

b. Edit A List

This operation will duplicate the form of the "*View a List*" option. The File Select window will contain the additional option "New File" and the Text Display window will become a Text Edit window allowing a user to edit the text within the window.

7. Coding / Decoding Functions Analysis

This category covers all the functionality set by the ATP-1 tactical signal coding/decoding requirements. The operations required to support this category are:

- Decode an ATP-1 Signal
- Encode an ATP-1 Signal

The two operations became the entries for the pulldown menu button ***Coding*** as shown in Figure 3-26. No further functionality was provided as the coding/decoding operations are not supported in the prototype.

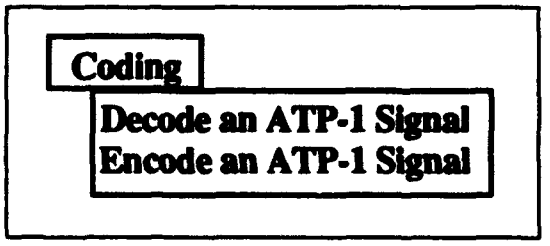

Figure 3-26. Coding Pulldown Options

8. System Input And Display Defaults Analysis

This category covers a wide range of required data input and system configuration issues. To keep the screen display manageable, only one operation was selected, with sub-menus as required to provide the necessary range of functionality.

- Set System Defaults

This operation, shown in Figure 3-2, leads the user to a series of menu windows which provide the interface into the additional operations required for this category.

- Set System Filters
- Set Alert Parameters
- Set External System Inputs
- Set Custom System Configurations

These five operations became the basis for the pulldown selection window shown if Figure 3-27.

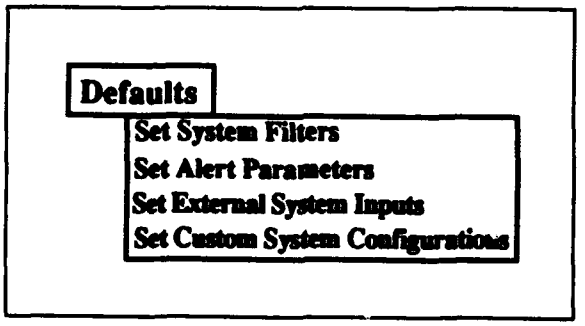

Figure 3-27. Defaults Pulldown Options

a. Set System Filters

There are two types of filters defined in the requirements, ***Tactical Database Filters*** and ***Screen Display Filters***. There are both conceptual and operational differences between the two filter types.

Database filters will affect the number of objects to be processed by the Integration system. Those objects not meeting the filter parameters will be ignored and remain unprocessed.

Display filters will affect only the number of objects to be displayed on the screen. The objects not meeting the filter parameters will not be displayed on the screen, but will continue to be processed by the Integration system ready for instant display. The Menu Bar option ***Filters***, as discussed in section B.6, allows the user to override the defined display filters, in which case all objects in the Tactical Database that meet the screen range scale limit will be passed to the ***Screen Display***.

The two filter types require two separate operations to implement.

- Set Tactical Database Filters
- Set Screen Display Filters

The two filter operations became the entries into another radio button select window as shown in Figure 3-28.

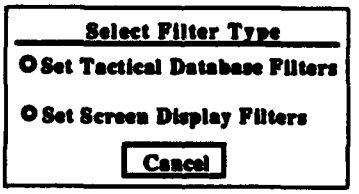

Figure 3-28. Select Filter Type

(1) Set Tactical Database Filters. The Tactical Database contains objects of varying types and attributes. Each object type can be filtered separately based on its attributes. For Track and Point type objects it is also possible to filter separately on categories of the object. The object types, categories and allowed filtering attributes are shown in Table 3-1.

To allow the user to make filtering choices within one composite screen, a partitioned checkbox window, shown in Figure 3-29 was constructed. The system default configuration has been selected for illustration purposes.

Selection of an item indicates that it should be included in the database processing. In the example shown, the colored items indicate a selection. The example will instruct the database to include all object types, of all categories and classifications. The blank Range, CPA, Class and Height boxes indicate that no filters for those items should be applied.

The And/Or selector buttons allow the user to define atomic filter entities, which will permit multiple filter attributes on a single object type. For example, if the user wishes to see surface friendlies within 30 nautical miles and surface hostiles within 50 nautical miles, the And selector would be employed to concatenate the filters within the single surface object type.

The Or selector allows filter concatenation within object types. If the user wishes to see all surface friendlies or all air hostiles or navigation hazards within 50 nautical miles, the Or selector would be used to define the multiple filter object types.

(2) Set Screen Display Filters. The Screen Display Filters are virtually identical to those of the Tactical Database. The one difference is that the Range parameter is set by the Range Scaling function and cannot be adjusted in this operation. A screen identical to Figure 3-29 will be presented to the user with all Range selection areas "grayed out", allowing no Range parameter inputs.

b. Set Alert Parameters

Alerts notify the user that there has been some condition or event detected that either places the ship in danger or adversely affects the operation of the system. There are five cases where an Alert is appropriate.

- Close CPA Detected
- Low Flyer Detected
- External System Status Changes
- Shallow Water Detected
- Unusual Sea Floor Ramping Detected

These five cases formed the entries for the checkbox window shown in Figure 3-30. The system defaults have been entered for illustration purposes.

(1) Alert on all close CPA situations. The system will alert the user if any Tracks are detected with CPA ranges less than the CPA alert value.

(2) Alert on all Low Flyer detections. The system will issue an alert for all cases of Air Tracks with a Height attribute less than the alert height value.

(3) Alert on all System status changes. The system will alert the user of all status changes in external system inputs.

Object Type	Categories	Filter Attributes
Track	Air	CPA Range Height Classification
Track	Surface Sub-Surface	CPA Range Classification
Point	Waypoint Reference Nav Hazard Man in Water	CPA Range
Area	---	CPA Range
Zone	---	CPA Range

Table 3-1. System Filter Table

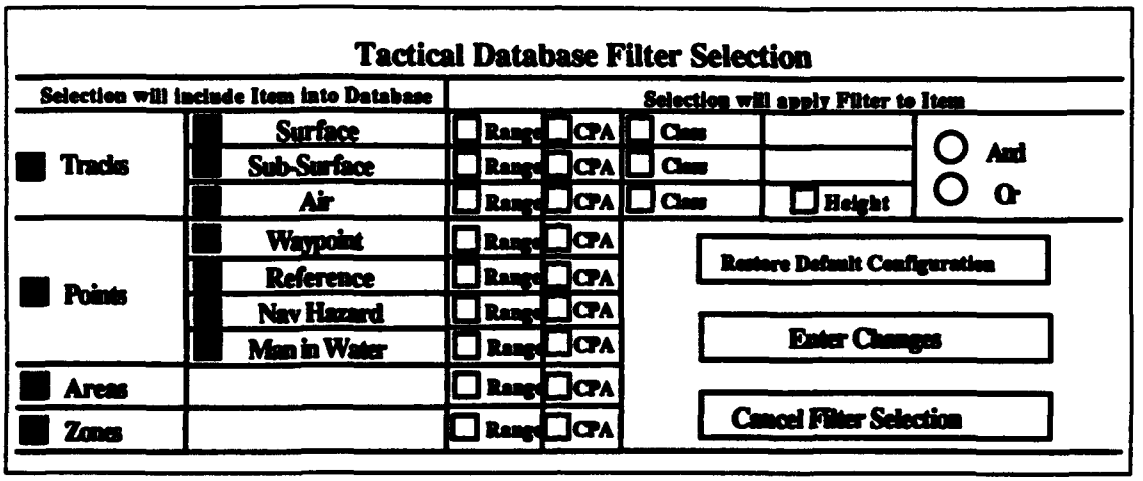

Figure 3-29. Database Filter Selection

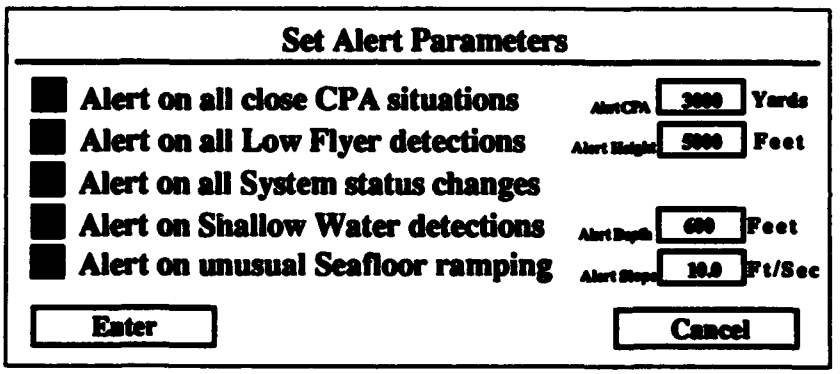

Figure 3-30. Set Alert Parameters

(4) Alert on all Shallow Water detections. The system will issue an alert when the fathometer inputs drop below the alert depth threshold.

(5) Alert on unusual Seafloor ramping. The fathometer inputs are monitored continuously and compared over time. Any drop in fathometer readings in excess of the alert slope threshold will result in an alert.

c. *Set External System Inputs*

The Tactical Information System cannot provide accurate displays without the information inputs external equipment provide. This operation allows the user to specify which units will act as information feeds into the system. It will also allow the user to designate equipment as inoperative, which will act as a reminder to overall ships equipment status.

All selections made in this area will be reflected in the System Status window located in the lower right corner of the display. Figure 3-31 is a checkbox window formed to implement this operation.

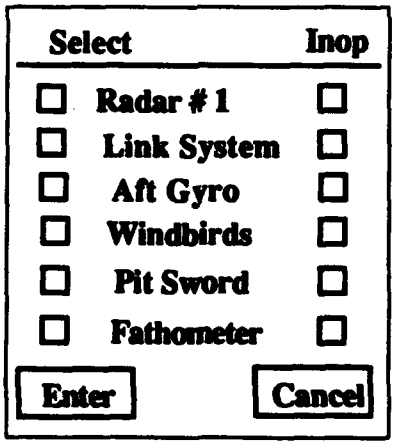

Figure 3-31. Equipment Select Window

d. Set Custom System Configurations

It is recognized that different individuals have varying preferences. This operation will allow a user to save a set of predefined system defaults and recall them at a later date. Figure 3-32 shows the radio button window that will appear when this option is selected.

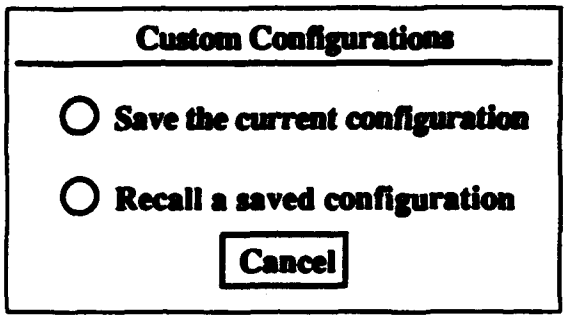

Figure 3-32. Custom Configurations

(1) Save the current configuration. This option will read in all configurable system options, with the exception of the external equipment selections, and save the information to a disk file. The user will be prompted for a configuration nameprior to the save. The option to cancel the operation will also be provided.

(2) Recall a saved configuration. This option will present the user with a scrolling text window containing the names of all the configurations currently in the system. Selecting a configuration will cause the system to reset all configurable items to the saved parameters.

9. Navigation Functions

This category encompasses all the requirements concerning path planning and Ownship navigation functions. There are four operations required for this option.

- Plot a Great Circle Path
- Override Navigation Equipment Inputs
- Plot an Intercept Course
- Plot an Avoidance Course

Figure 3-33 is a radio button window containing the four operations defined for this category.

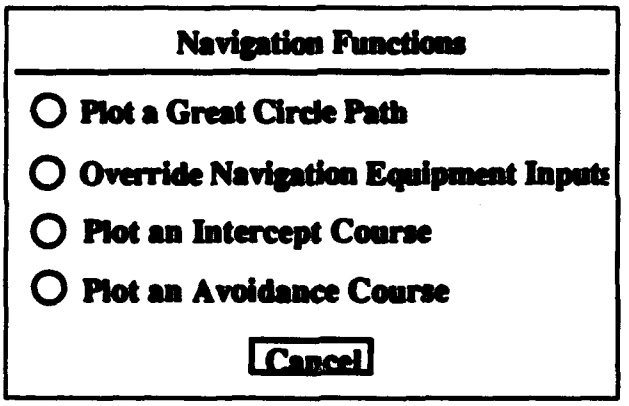

Figure 3-33. Navigation Functions

a. Plot A Great Circle Path.

It is often the case that shipboard navigation officers must plot courses from one place to another by this method. This capability is relatively easy to implement based on the components and types that are already part of the prototype system. This option was included for future development as no Path plotting capability is planned for the prototype.

b. Override Navigation Equipment Inputs.

Not all equipment is as reliable as one desires. This operation was inserted to allow the user to place any and all Navigation systems in "Manual" mode, and make all Navigation inputs from the keyboard.

All manual inputs made from this option will pass into the Tactical Database and replace the automatic Navigation equipment inputs. A message, "Navigation: Manual", will be displayed in the System Status window alerting the user that at least one navigation input is in manual mode. Figure 3-34 is the checkbox option window designed for this operation.

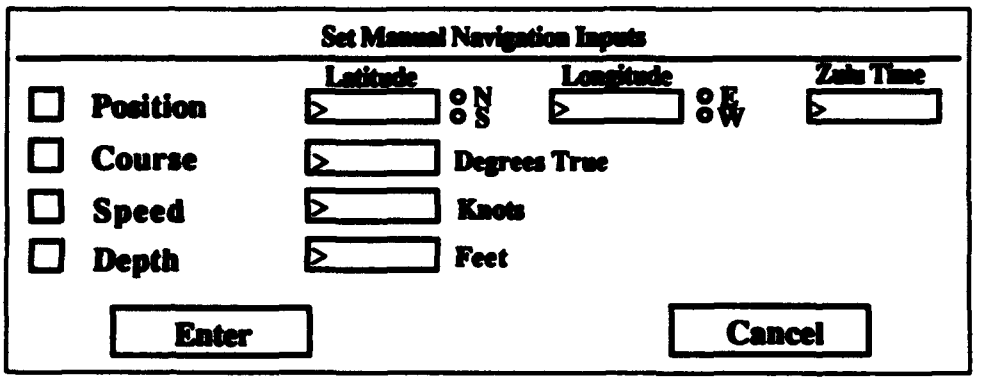

Figure 3-34. Set Manual Inputs

c. Plot An Intercept Course.

This option will perform the moving geometry calculations required to determine a course and speed from one object to intercept another. The user will be prompted, (with the Warning Select window shown in Figure 3-20), to select two screen

objects. The calculations will then be performed and the results displayed in the Recommendations display window.

d. Plot An Avoidance Course.

This option is the conceptual opposite of intercepting an object. The system will compute the necessary course and speed to avoid an object by a specified distance. The user will be prompted to select two screen objects and then asked for an avoidance distance. The system will display the results of the ensuing calculation in the Recommendations display window.

C. INFORMATION DISPLAY ANALYSIS

In this phase of the design all categories, requirements, operations and other reference documents were examined for items of information that were either required or available for display. A large list of information items was compiled and then split into two sections.

The first section of the list contained information items that, based on the system requirements, were required to be displayed. Items such as Track and Point objects, menus, recommendations and alerts made up the majority of this list.

The second section contained items that were not specifically called out in the requirements, but were available in the system. These "nice to have" items consisted of configuration and system status selections, time and external equipment inputs.

The required items list was studied with the objective of minimizing the number of display windows needed to present all information elements. This analysis resulted in the definition of six primary information windows and two menu windows.

- Graphical Track Display Window
- Track Information Window
- System Alerts Window
- System Recommendations Window
- System Intelligence Window
- System Status Window
- System Menu Bar
- Title Window

The decision regarding which elements of "nice to have" information to include in the design was postponed pending the final design of the required windows.

1. The Graphical Track Display Window

This, the largest of the information display windows, contains graphical representations of all track objects defined in the system. Tracks are displayed in the circular area as standard NTDS symbols, identified by the text appearing below the symbols. The circular display closely approximates standard radar repeater form and was chosen to ease the users transition from standard shipboard systems to the Information Display.

The circular display also provides usable space to provide many elements of the "nice to have" information in a clean and easy to read format. Computed information, such as True Wind speed and direction and Set direction and Drift speed were included as were the direct system inputs of Depth and Time. Additional system status information

was included to provide the user with the current default values of system filters, alerts and intel.

The capability to scale the range display to user defined dimensions was included as a slider bar. Each selection of the range scaling function will double/halve the current scale selected to a minimum of 2000 yards and a maximum of 512 nautical miles. Figure 3-35 shows a representation of the final design including a sample of typical system outputs.

2. The Track Information Window

This window required a text display of all predefined object attributes as well as computed attributes such as CPA. As the display was to be text based, a scrolling text window was selected as the interface. Figure 3-36 shows the final design form and a selection of typical information items found in the window.

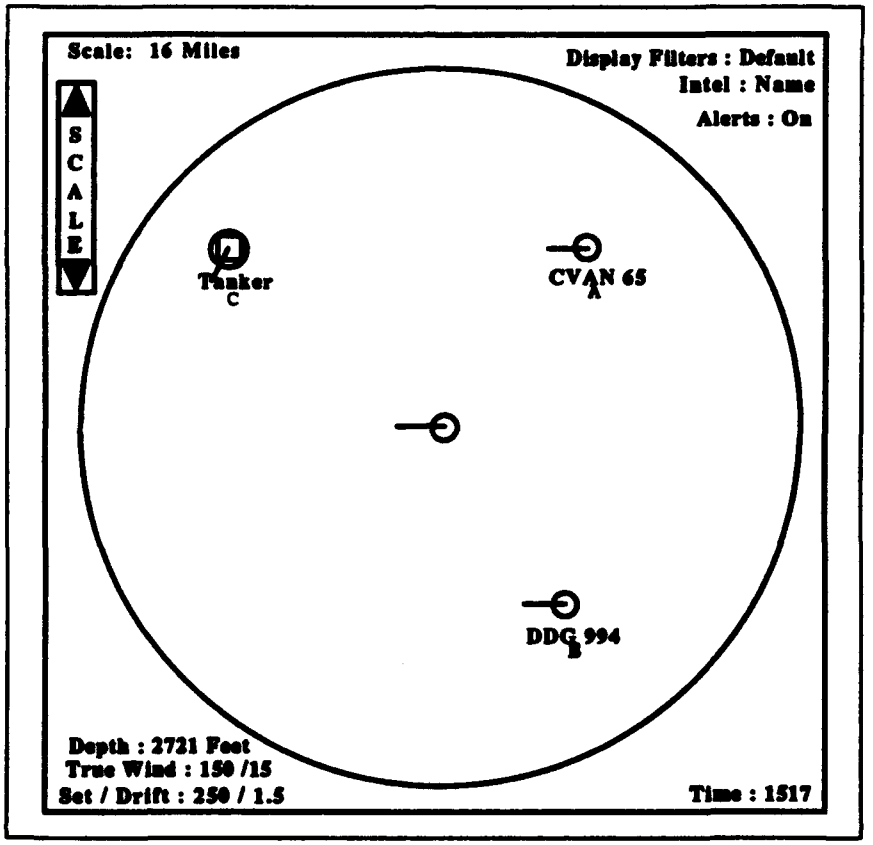

Figure 3-35. The Graphical Track Display Window

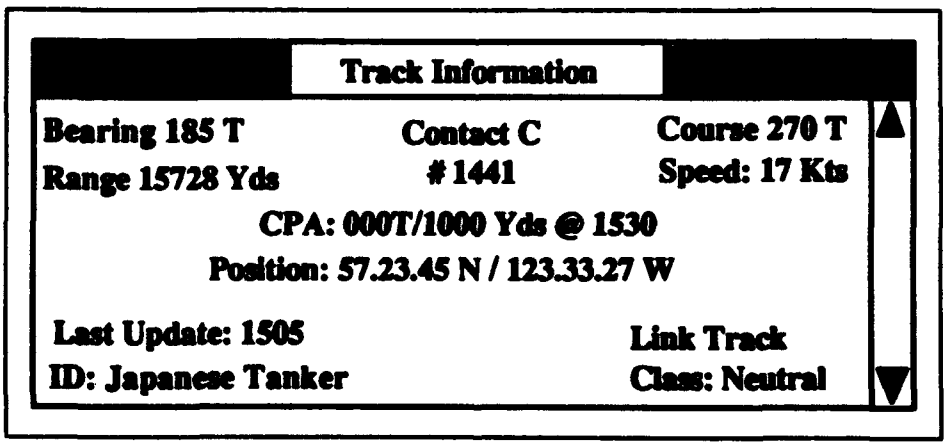

Figure 3-36. Track Information

This window design serves track objects of all types. The title **Track Information** was selected after a considerable amount of discussion with the proposed end-users of this system.

The proposed end-user, an American Naval Officer, was found to be more comfortable with the term Track vs. the term Object. As the field of Object Oriented Design is still in its infancy, and not all users could be expected to understand what an object meant, the title Track Information was adopted.

Another benefit of the discussions with the end users was the discovery that most informational reports, generally passed verbally, are formatted fairly specifically. The verbal reports generally begin with the name of the object, followed by the group bearing and range, and concluding with the group course, speed and CPA.

The placement of information within this window attempts to duplicate the verbal report format, grouping the bearing and range attributes into one "block" and the CPA is prominently featured in the middle of the window. The other attributes, judged less critical by the end-users, were placed at the bottom of the window. course and speed attributes into another. The name is centered in the top of the display.

3. The Systems Alerts Window

This window displays warnings to the user based on the alert parameters defined in the **Set Alert Parameters** section of the **System Defaults**. The display is scrollable to allow the user to review all alerts generated by the system during the current mission. Figure 3-37 shows the final window design.

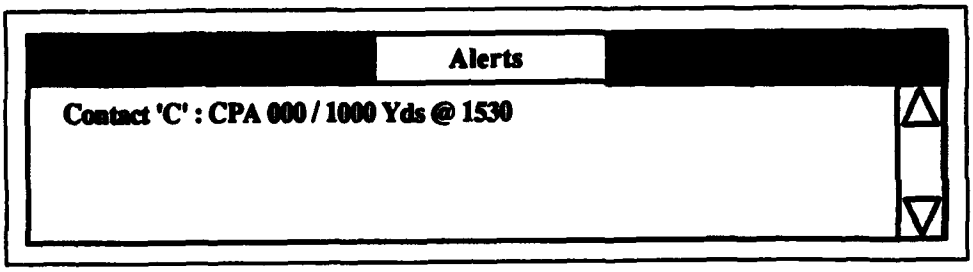

Figure 3-37. System Alerts Window

4. The System Recommendations Window

This window provides course and speed recommendations to the user based on the alert threshold defined in the System Defaults section. The computed course and speed necessary to avoid any track by the alert threshold will be displayed in this window, paired with each CPA alert in the alerts window. Figure 3-38 shows the window with a typical recommendation.

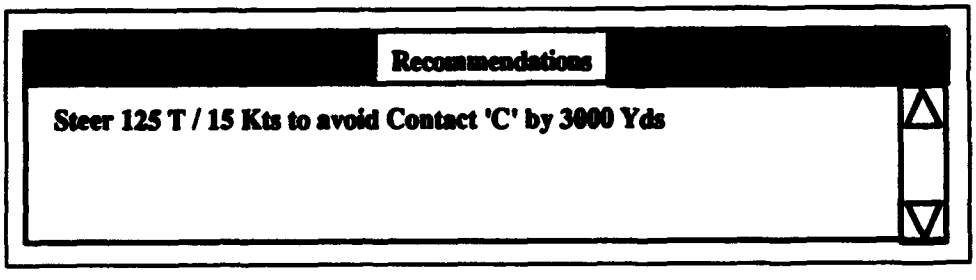

Figure 3-38. The System Recommendations Window

5. The System Intelligence Window

This window, provided for future development, will present all pertinent Intel information available for the selected track. Figure 3-39 shows a projected representation of the final design.

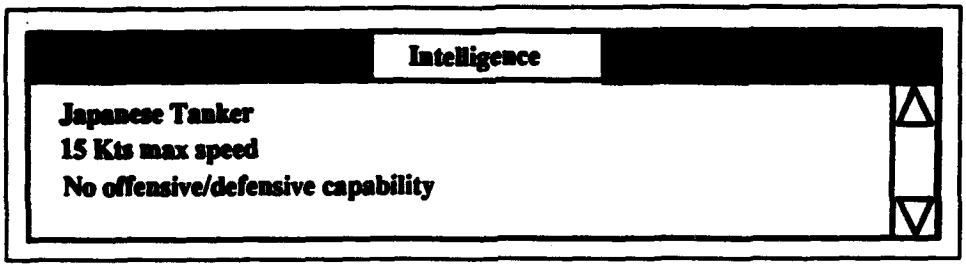

Figure 3-39. The System Intelligence Window

6. The System Status Window

This window presents the user with a comprehensive view of the current status of all system inputs. Reverse video is employed to distinguish degraded or inoperative system inputs to the user. Figure 3-40 depicts the window containing a set of system inputs typical to the system.

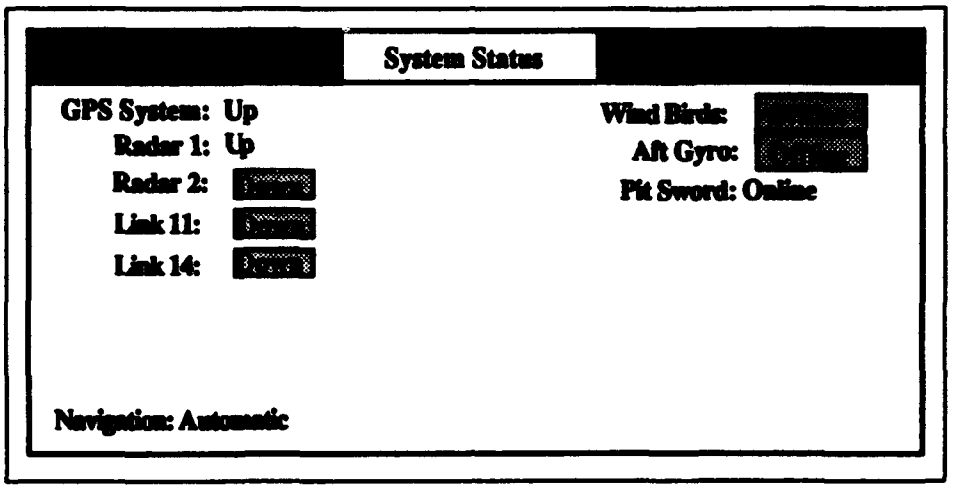

Figure 3-40. The System Status Window

7. The System Menu Bar

This Menu bar presents the user with the nine pulldown menu operations to exercise the different capabilities of the Information System. Figure 3-41 shows the final design of the Menu bar, reprinted from Figure 3-1 shown earlier.

Figure 3-41. The System Menu Bar

8. The Title Window

The Title window serves a dual purpose. As the final system design encompasses the entire screen display, some area of the screen was needed to present the sub-menu items. This window was found to be the ideal position for all sub-menus and intermediate information presentations. A select button was added into this window to

allow the user to exit cleanly from the system. Figure 3-42 shows the window in its final design form, reprinted from Figure 3.2 presented earlier.

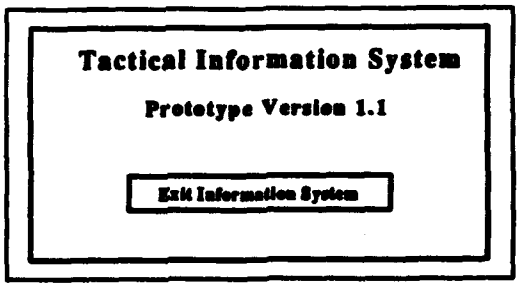

Figure 3-42. The Title Window

D. WINDOW SIZING AND POSITIONAL ANALYSIS

1. The Prototype Display Model

This, the last design stage, required the sizing and positioning of the different windows and menu items in a manner easily readable at a glance by the user. The final design is shown in Figure 3-43.

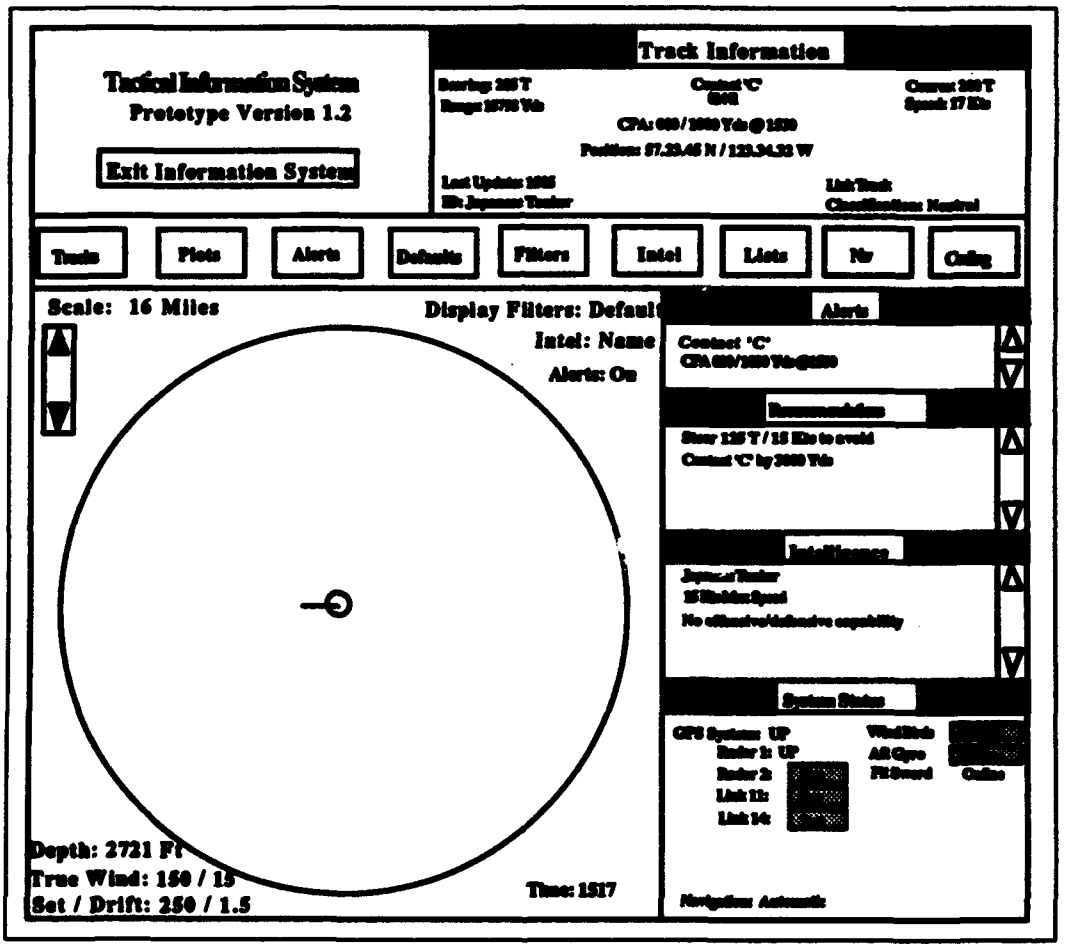

Figure 3-43. The Prototype Display

IV. INTERFACE IMPLEMENTATION

A. ADA TOOLS AND GRAPHICS SUPPORT REVIEW

One of the research goals defined for this thesis was to ascertain if a viable graphic user interface could be completed in Ada. The implementation phase of this project was started with a review of the tools available to port Ada into X11R4. The number of those available tools is very small; only two products currently exist.

1. TAE+

Transportable Applications Environment Plus (TAE+) was developed by the National Aeronautics and Space Administration (NASA) on behalf of the Goddard Space Flight Center. TAE+ is a complete programming and design package which supports several widgets (display types, such as slider bars, text boxes, buttons, check boxes, etc.) and is capable of generating Ada code directly from a defined screen application.

A complete description of the TAE+ package is available in [TAE90].

2. STARS

The Software Technology for Adaptable, Reliable Systems (STARS) program is an ongoing project sponsored by the U.S. Air Force to provide Ada programming support in the form of Ada libraries and toolkits. The Ada/Xlib libraries provided by STARS give an Ada programmer direct access to all low level X Protocol primitives. An

additional STARS tool, providing an Ada/Xt widget toolkit and an Ada/Motif widget toolkit are under development and will be available soon.

B. PROGRAMMING THE INTERFACE

The interface was completed as a set of six Ada packages, dividing the necessary functionality into logical groups.

- Open Interface Displays Package
- Monitor Displays Package
- Process Interface Displays Package
- Tacplot Set Package
- Draw Display Graphics Package
- TAE Menu/Main Program

1. Open Interface Displays Package

This package contains all the declarations and parameters required to open each of the six information display windows defined in the design. A separate procedure was included for each window type. Complete program listings are available in Appendix A.

2. Monitor Displays Package

This package contains one monitor task and one test procedure. Complete program listings are available in Appendix B.

a. Monitor Update Intervals Task

This task simply keeps track of the amount of time elapsed between screen refreshes. The timing requirements in [SS90] require a maximum of four second delays between screen updates so the task will call the process to refresh the screen every four seconds.

b. Test System Status Procedure

This infrequently called procedure is initiated by pressing the mouse button in the System Status display window. This procedure polls the Bolick/Irwin Status package to determine current system status and displays that information within the System Status window.

3. Process Interface Displays Package

This package contains procedures and functions which process information related to the screen displays. Complete program listings are available in Appendix C.

a. Process Tacplot Objects Procedure

This procedure takes each object defined in the Tactical Plot Set Package and computes all the required elements of information to be displayed on the screen. Each object is processed individually and the screen drawing and writing processes are called to project the information onto the display screen.

b. Process Hook Position

Each time the mouse button is pressed within the Tactical Plot window, this procedure is called to determine the position of the cursor at the time the button was

pressed. Various screen positions are predefined as either ownship or screen scaling functions and this procedure checks to see if the mouse button was pressed in one of those locations.

If the mouse was not pressed in a predefined location, a state variable is set and the location of each incoming object is compared to the "hook" position to determine if an object was in the vicinity of the hook location. If the values of the hook position and the location of any track object correspond, the hook will be displayed surrounding the object.

4. Draw Display Graphics Package

This package contains all the screen drawing and writing procedures for both object and information displays. The symbology used to display the track objects was taken from [GOTS91]. Complete program listings are provided in Appendix D.

5. Tacplot Set Package

This package is the primary link between the Bolick/Irwin Integration System and the display process. A generic set package was used to implement a track data structure and procedures were provided to empty the set, fill the set and retrieve the data. Each data object in the set is sent to the Process Displays package individually for processing prior to display on the screen. Full program listings are available in Appendix E.

6. Menu Package

This package contains the code generated by TAE+, heavily modified to implement the menuing system described in Chapter III. The system starts by opening all windows defined by both TAE+ and STARS, loading an Ownship track object and 3 test tracks, and entering an endless event monitoring loop. Complete program listings are available in Appendix F.

C. INTERFACE TESTING

1. System Integration

Upon completion of each program module package, the module was linked into the Bolick/Irwin Tactical Database project and tested both independently and with the other Interface packages. Informal timing tests were conducted at each stage to detect any unacceptable system response times.

2. System Timing Results

Timing tests were completed for the packages containing the STARS code meeting with great success. The package containing the TAE+ code initially provided unacceptable system response times when integrated with the system as a whole, and was successfully modified to give acceptable response results.

a. STARS Response Times

With the STARS code fully integrated into the system, the response times for an object base of four tracks given five rapid button presses averaged less than 0.1

seconds per display update. An object base of 20 tracks was provided with the results remaining essentially identical.

b. TAE+ Response Times

With the TAE+ code integrated unmodified into the system, the response times for the same test of four track objects given five rapid button clicks averaged three seconds per display update. An object base of 20 track was provided with the timing results degenerating to four seconds per display update.

TAE+ code is generated to run as a main procedure. The code contains a display opening procedure, an event monitoring loop and a support package containing the actions to be taken when an event is detected.

The event monitoring loop is constructed as a single IF-THEN clause which monitors every object on every defined display once per loop cycle. This inefficient method of event monitoring must be modified before acceptable timing results can be obtained. Placing the event monitoring loop into a task structure is not possible as TAE+ uses the Ada pragma Interface to access the X11 functions.

The modifications made to the TAE+ code included changing the single IF-THEN clause to a nested structure, which monitors only the defined windows, adding a top level IF-THEN clause, (using the TAE+ directive "WPT_PENDING"), to isolate the TAE+ event loop unless an event has taken place in a TAE+ window and placing all STARS event handlers into the same loop as the TAE+ event handlers.

These changes resulted in timing results similar to those recorded for the STARS package alone. The speed at which TAE+ opens and closes windows is noticibly slower than STARS but is transparent to the display updating process due to the changes made to the event handling loop.

V. CONCLUSIONS

A. SUMMARY

This thesis presented the design and implementation of a graphical user interface for a near-real-time system. Two Ada graphical interface tools were tested and evaluated for acceptable system response times.

1. STARS

The STARS Xlib bindings are a wonderful tool for building windows, line art, pixmaps and static text displays. There are, however, no provisions for any standard widget types such as scroll bars, text input boxes, view ports or dialog boxes. The implementation of an Ada application using STARS is quite straightforward, although knowledge of X Protocol primitives is very helpful to a novice programmer.

The Xlib bindings are extremely fast and highly suitable for a real time application.

2. TAE+

The TAE+ application generator is a very useful tool for any non-time critical application. All standard widget types are supported and the user interface into the program is very simple.

Constraints can be placed on all data input streams and are checked by TAE+ at execution time, relieving the programmer of the task of building constraint che

routines. The execution time of the program can be improved by reducing the number of defined windows (called panels in TAE+) and by placing the window event handlers into a nested IF-THEN clause with the TAE function WPT_PENDING used as an isolation device. However, even with the mentioned modifications to the TAE+ code, the speed at which TAE+ opens and closes display windows makes the execution time of the generated program application is comparatively slow, and unacceptable for a real-time graphical information display application. However, for rapidly prototyping menuing functions which link into a fast display process such as the STARS XLib library, it is an invaluable tool.

The Ada to X11 bindings for TAE+ are different from those used by STARS, and when used in conjunction with a STARS application, results in linking two separate sets of Ada to X11 bindings into the final executable. This dual linking effectively increased the size of the final Interface program by a factor of approximately 2.5 (1.2 megabytes to 2.9 megabytes) which may affect the transportablity of the final application dur to memory constraints.

TAE+ currently has an upgraded package (Version 4.2) under test and many of the problems discussed in this chapter may have been addressed in the new edition.

B. RECOMMENDATIONS

Any graphical user interface requiring text or numeric data input requires a number of different widget types to present the user with an easily understood interface. A widget toolkit, called Ada/Xt, for Ada applications has recently been introduced by STARS as an "A" test release and is available via anonymous FTP from host "stars.rosslyn.unisys.com" in the "pub" directory. This widget toolkit runs on top of the Ada/Xlib bindings and is configured for the VADS or Telesoft Ada compilers. Due to the display speed constraints encountered in the TAE+ package, I recommend that any further research into Ada user interfaces obtain and test this new STARS package for interface development. Full details of the Ada/Xt widget toolkit are available in [ADAXT].

C. SUMMARY OF CONTRIBUTION

This project was completed entirely in the Ada programming language, and employed a number of tools representing the current "state of the art" in Ada graphics processing. Preliminary research into Ada graphics applications produced no data on any other programming attempts of this scale.

The design approach and the implementation methods used for the project are entirely transparent and can be applied to any other user interface project using Ada.The

modifications made to the TAE+ generated code, and the methods used to link that code into a larger application is entirely portable for any application using the TAE+ package.

APPENDIX A

OPEN INTERFACE DISPLAYS

```
--        Author: Michael G. Stockwell
--        Date  : 21 September 1991
--        Description: Handles the declarations and opening procedures for
                       all windows defined in the LCCDS program

with System; with Key_Syms; with Text_Io; with X_Lib;

package Open_Interface_Displays is
   use X_Lib;
   use Events;
-- General Declarations
   Display_Not_Open          : Exception;

-- Declarations for the Tactical Plot Window

   Tacplot_Buffer            : X_Lib.String_Pointer := new String'(" ");
   Tacplot_Gc_Value_Mask     : X_Lib.Graphic_Output.Gc_Mask_Type
                                          := Graphic_Output.Gc_Font;
   Tacplot_Gc_Value_Rec      : X_Lib.Graphic_Output.Gc_Value_Record;
   Tacplot_Main_Attrs        : X_Lib.Set_Window_Attributes_Record;
   Tacplot_Main_Screen       : X_Lib.Screen_Number;
   Tacplot_Window            : X_Lib.Window;
   Tacplot_Root_Wind         : X_Lib.Window;
   Tacplot_Border_Width      : X_Lib.Coordinate      := 2;
   Tacplot_Drawing_Space     : X_Lib.Drawable;
```

```
   Tacplot_Display           : X_Lib.Display;
   Tacplot_Display_Context   : X_Lib.Graphic_Output.Graphic_Context;
   Tacplot_Window_X          : X_Lib.Coordinate := 0;
   Tacplot_Window_Y          : X_Lib.Coordinate := 0;
   Tacplot_Window_Width      : X_Lib.Pixel := 700;
   Tacplot_Window_Height     : X_Lib.Pixel := 700;
   Tacplot_Status            : X_Lib.Events.Compose_Status_Record;
   Tacplot_Wa_Values         : X_Lib.Wa_Mask_Type:= Wa_Background_Pixel or
                                                    Wa_Border_Pixel or
                                                    Wa_Cursor or
                                                    Wa_Override_Redirect or
                                                    Wa_Event_Mask or
                                                    Wa_Bit_Gravity or
                                                    Wa_Win_Gravity;

-- Declarations for the Track Information Window

   Track_Info_Buffer         : X_Lib.String_Pointer := new String'(" ");
   Track_Info_Display        : X_Lib.Display;
   Track_Info_Drawing_Space: X_Lib.Drawable;
   Track_Info_Gc_Value_Mask: X_Lib.Graphic_Output.Gc_Mask_Type
                                              := Graphic_Output.Gc_Font;
   Track_Info_Gc_Value_Rec : X_Lib.Graphic_Output.Gc_Value_Record;
   Track_Info_Border_Width : X_Lib.Coordinate      := 2;
   Track_Info_Display_Context : X_Lib.Graphic_Output.Graphic_Context;
   Track_Info_Window_X             : X_Lib.Coordinate      := 0;
   Track_Info_Window_Y             : X_Lib.Coordinate      := 0;
   Track_Info_Window_Width         : X_Lib.Pixel           := 600;
   Track_Info_Window_Height        : X_Lib.Pixel           := 102;
   Track_Info_Main_Attrs           : X_Lib.Set_Window_Attributes_Record;
   Track_Info_Main_Screen          : X_Lib.Screen_Number;
```

```
   Track_Info_Window              : X_Lib.Window;
   Track_Info_Root_Wind           : X_Lib.Window;
   Track_Info_Status              : X_Lib.Events.Compose_Status_Record;
   Track_Info_Wa_Values           : X_Lib.Wa_Mask_Type:=
                                          Wa_Background_Pixel or
                                          Wa_Border_Pixel or
                                          Wa_Cursor or
                                          Wa_Override_Redirect or
                                          Wa_Event_Mask or
                                          Wa_Bit_Gravity or
                                          Wa_Win_Gravity;

-- Declarations for the Alerts Info Window

   Alerts_Buffer           : X_Lib.String_Pointer := new String'(" ");
   Alerts_Gc_Value_Mask    : X_Lib.Graphic_Output.Gc_Mask_Type
                                    := Graphic_Output.Gc_Font;
   Alerts_Gc_Value_Rec     : X_Lib.Graphic_Output.Gc_Value_Record;
   Alerts_Main_Attrs       : X_Lib.Set_Window_Attributes_Record;
   Alerts_Main_Screen      : X_Lib.Screen_Number;
   Alerts_Window           : X_Lib.Window;
   Alerts_Root_Wind        : X_Lib.Window;
   Alerts_Border_Width     : X_Lib.Coordinate     := 2;
   Alerts_Drawing_Space    : X_Lib.Drawable;
   Alerts_Display          : X_Lib.Display;
   Alerts_Display_Context  : X_Lib.Graphic_Output.Graphic_Context;
   Alerts_Window_X         : X_Lib.Coordinate     := 0;
   Alerts_Window_Y         : X_Lib.Coordinate     := 0;
   Alerts_Window_Width     : X_Lib.Pixel          := 433;
   Alerts_Window_Height    : X_Lib.Pixel          := 90;
   Alerts_Status           : X_Lib.Events.Compose_Status_Record;
```

```
Alerts_Wa_Values          : X_Lib.Wa_Mask_Type:= Wa_Background_Pixel or
                                                 Wa_Border_Pixel or
                                                 Wa_Cursor or
                                                 Wa_Override_Redirect or
                                                 Wa_Event_Mask or
                                                 Wa_Bit_Gravity or
                                                 Wa_Win_Gravity;
-- Declarations for the Recommendations Info Window

Recommendations_Buffer            : String_Pointer := new String'(" ");
Recommendations_Gc_Value_Mask     : X_Lib.Graphic_Output.Gc_Mask_Type
                                  := Graphic_Output.Gc_Font;
Recommendations_Gc_Value_Rec      : Graphic_Output.Gc_Value_Record;
Recommendations_Main_Attrs        : Set_Window_Attributes_Record;
Recommendations_Main_Screen       : X_Lib.Screen_Number;
Recommendations_Window            : X_Lib.Window;
Recommendations_Root_Wind         : X_Lib.Window;
Recommendations_Border_Width      : X_Lib.Coordinate      := 2;
Recommendations_Drawing_Space     : X_Lib.Drawable;
Recommendations_Display           : X_Lib.Display;
Recommendations_Display_Context   : Graphic_Output.Graphic_Context;
Recommendations_Window_X          : X_Lib.Coordinate      := 0;
Recommendations_Window_Y          : X_Lib.Coordinate      := 0;
Recommendations_Window_Width      : X_Lib.Pixel           := 433;
Recommendations_Window_Height     : X_Lib.Pixel           := 90;
Recommendations_Status            : Events.Compose_Status_Record;
Recommendations_Wa_Values         : X_Lib.Wa_Mask_Type:=
                                          Wa_Background_Pixel or
                                          Wa_Border_Pixel or
                                          Wa_Cursor or
                                          Wa_Override_Redirect or
```

```
                                                Wa_Event_Mask or
                                                Wa_Bit_Gravity or
                                                Wa_Win_Gravity;
-- Declarations for the Intel Info Window

   Intel_Buffer            : X_Lib.String_Pointer := new String'(" ");
   Intel_Gc_Value_Mask     : X_Lib.Graphic_Output.Gc_Mask_Type
                                          := Graphic_Output.Gc_Font;
   Intel_Gc_Value_Rec      : X_Lib.Graphic_Output.Gc_Value_Record;
   Intel_Main_Attrs        : X_Lib.Set_Window_Attributes_Record;
   Intel_Main_Screen       : X_Lib.Screen_Number;
   Intel_Window            : X_Lib.Window;
   Intel_Root_Wind         : X_Lib.Window;
   Intel_Border_Width      : X_Lib.Coordinate      := 2;
   Intel_Drawing_Space     : X_Lib.Drawable;
   Intel_Display           : X_Lib.Display;
   Intel_Display_Context   : X_Lib.Graphic_Output.Graphic_Context;
   Intel_Window_X          : X_Lib.Coordinate      := 0;
   Intel_Window_Y          : X_Lib.Coordinate      := 0;
   Intel_Window_Width      : X_Lib.Pixel           := 433;
   Intel_Window_Height     : X_Lib.Pixel           := 175;
   Intel_Status            : X_Lib.Events.Compose_Status_Record;
   Intel_Wa_Values         : X_Lib.Wa_Mask_Type:= Wa_Background_Pixel or
                                                Wa_Border_Pixel or
                                                Wa_Cursor or
                                                Wa_Override_Redirect or
                                                Wa_Event_Mask or
                                                Wa_Bit_Gravity or
                                                Wa_Win_Gravity;
```

```
-- Declarations for the System_Status Info Window

   System_Status_Buffer            : String_Pointer := new String'(" ");
   System_Status_Gc_Value_Mask     : X_Lib.Graphic_Output.Gc_Mask_Type
                                          := Graphic_Output.Gc_Font;
   System_Status_Gc_Value_Rec      : Graphic_Output.Gc_Value_Record;
   System_Status_Main_Attrs        : X_Lib.Set_Window_Attributes_Record;
   System_Status_Main_Screen       : X_Lib.Screen_Number;
   System_Status_Window            : X_Lib.Window;
   System_Status_Root_Wind         : X_Lib.Window;
   System_Status_Border_Width      : X_Lib.Coordinate      := 2;
   System_Status_Drawing_Space     : X_Lib.Drawable;
   System_Status_Display           : X_Lib.Display;
   System_Status_Display_Context   : Graphic_Output.Graphic_Context;
   System_Status_Window_X          : X_Lib.Coordinate      := 0;
   System_Status_Window_Y          : X_Lib.Coordinate      := 0;
   System_Status_Window_Width      : X_Lib.Pixel           := 433;
   System_Status_Window_Height     : X_Lib.Pixel           := 254;
   System_Status_Status            : X_Lib.Events.Compose_Status_Record;
   System_Status_Wa_Values         : X_Lib.Wa_Mask_Type:=
                                          Wa_Background_Pixel or
                                          Wa_Border_Pixel or
                                          Wa_Cursor or
                                          Wa_Override_Redirect or
                                          Wa_Event_Mask or
                                          Wa_Bit_Gravity or
                                          Wa_Win_Gravity;

   procedure Open_Tacplot_Display;
   procedure Open_Track_Info_Display;
   procedure Open_Intel_Display;
```

```
   procedure Open_System_Status_Display;
   procedure Open_Alerts_Display;
   procedure Open_Recommendations_Display;

end Open_Interface_Displays;
-------------------------------------------------------------------------
package body Open_Interface_Displays is
   procedure Open_Tacplot_Display is
     begin

   -- Opening the display, get the new Tacplot_Display, get the default
   -- screen, set up the bounds of the panel, get the root window id
   -- and then create a window with our new defaults for the screen.
   --
   Tacplot_Display := X_Lib.X_Open_Display ("");

      if (Tacplot_Display = X_Lib.Null_Display) then
         raise Display_Not_Open;
      end if;

   Tacplot_Main_Screen := X_Lib.Default_Screen (Tacplot_Display);

   Tacplot_Root_Wind := Root_Window (Tacplot_Display,
                                     Tacplot_Main_Screen);
   Tacplot_Main_Attrs.Current_Cursor    := Cursors.X_Create_Font_Cursor
                               (Tacplot_Display, X_Lib.Cursors.Xc_Cross);
   Tacplot_Main_Attrs.Border_Pixel      := X_Lib.Black_Pixel
                              (Tacplot_Display, Tacplot_Main_Screen);

   Tacplot_Main_Attrs.Background_Pixel := 8;
   Tacplot_Main_Attrs.Bit_Gravity       := Center_Gravity;
```

```
Tacplot_Main_Attrs.Window_Gravity    := Center_Gravity;
Tacplot_Main_Attrs.Event_Mask        := Events.Exposure_Mask or
                                        Events.Button_Press_Mask or
                                        Events.Structure_Change_Mask;

--
-- With many of the parameters that we have just populated
-- we now create the window. The parameter Tacplot_Window will
-- be the new window_id associated with this new window.
--

Tacplot_Window :=
   X_Lib.X_Create_Window
       (Tacplot_Display,
        Tacplot_Root_Wind,
        Tacplot_Window_X,
        Tacplot_Window_Y,
        Tacplot_Window_Width,
        Tacplot_Window_Height,
        Tacplot_Border_Width,
        X_Lib.Default_Depth (Tacplot_Display, Tacplot_Main_Screen),
        X_Lib.Input_Output,
        X_Lib.Default_Visual (Tacplot_Display, Tacplot_Main_Screen),
        Tacplot_Wa_Values,
        Tacplot_Main_Attrs);

  Events.X_Select_Input (Tacplot_Display, Tacplot_Window,
                         Tacplot_Main_Attrs.Event_Mask);
X_Set_Icon_Name (Tacplot_Display, Tacplot_Window, "Tacplot");
X_Store_Name (Tacplot_Display, Tacplot_Window, "Tactical Plot");
X_Lib.Cursors.X_Define_Cursor (Tacplot_Display, Tacplot_Window,
```

```
                                    Tacplot_Main_Attrs.Current_Cursor);

Tacplot_Drawing_Space := Drawable (Tacplot_Window);

--
-- Here we set the text font to a 8X13 bold size and
-- create a Graphic_Context for the specified drawable
-- window.
--
Tacplot_Gc_Value_Rec.Font_Id := Fonts.X_Load_Font
                                    (Tacplot_Display, "8x13bold");

Tacplot_Display_Context := Graphic_Output.X_Create_Gc
                                                (Tacplot_Display,
                                                 Tacplot_Drawing_Space,
                                                 Tacplot_Gc_Value_Mask,
                                                 Tacplot_Gc_Value_Rec);
--
-- Now we are going to Map the Tacplot_Window to the specified display
--
X_Lib.X_Map_Window (Tacplot_Display, Tacplot_Window);
X_Lib.Events.X_Flush (Tacplot_Display);
--
-- This sets the text foreground color to be white
-- and the background color to be blue
--

X_Lib.Graphic_Output.X_Set_Background
                        (Tacplot_Display, Tacplot_Display_Context, 8);
X_Lib.Graphic_Output.X_Set_Foreground
                        (Tacplot_Display, Tacplot_Display_Context, 0);
```

```
exception

   when Display_Not_Open  =>
      Text_Io.Put_Line ("Could not open Display");
end Open_Tacplot_Display;


   procedure Open_Track_Info_Display is
     begin
  -- Opening the display, get the new Track_Info_Display, get the default
   -- screen, set up the bounds of the panel, get the root window id
   -- and then create a window with our new defaults for the screen.
   --
   Track_Info_Display := X_Lib.X_Open_Display ("");

      if (Track_Info_Display = X_Lib.Null_Display) then
         raise Display_Not_Open;
      end if;

   Track_Info_Main_Screen := X_Lib.Default_Screen (Track_Info_Display);

   Track_Info_Root_Wind := X_Lib.Root_Window (Track_Info_Display,
                                              Track_Info_Main_Screen);
   Track_Info_Main_Attrs.Current_Cursor:= Cursors.X_Create_Font_Cursor
                                          (Track_Info_Display,
                                           X_Lib.Cursors.Xc_Box_Spiral);
   Track_Info_Main_Attrs.Border_Pixel     := 12;
   Track_Info_Main_Attrs.Background_Pixel := 7;
   Track_Info_Main_Attrs.Bit_Gravity      := Center_Gravity;
   Track_Info_Main_Attrs.Window_Gravity   := Center_Gravity;
```

```
Track_Info_Main_Attrs.Event_Mask      := Events.Structure_Change_Mask;
Track_Info_Window :=
   X_Lib.X_Create_Window
       (Track_Info_Display,
        Track_Info_Root_Wind,
        Track_Info_Window_X,
        Track_Info_Window_Y,
        Track_Info_Window_Width,
        Track_Info_Window_Height,
        Track_Info_Border_Width,
        X_Lib.Default_Depth (Track_Info_Display,
                             Track_Info_Main_Screen),
        X_Lib.Input_Output,
        X_Lib.Default_Visual (Track_Info_Display,
                              Track_Info_Main_Screen),
        Track_Info_Wa_Values,
        Track_Info_Main_Attrs);
Events.X_Select_Input (Track_Info_Display, Track_Info_Window,
                       Track_Info_Main_Attrs.Event_Mask);
X_Set_Icon_Name (Track_Info_Display, Track_Info_Window,
                 "Track_Info");
X_Store_Name (Track_Info_Display, Track_Info_Window,
              "Track Information");

X_Lib.Cursors.X_Define_Cursor (Track_Info_Display, Track_Info_Window,
                               Track_Info_Main_Attrs.Current_Cursor);
Track_Info_Drawing_Space := Drawable (Track_Info_Window);
Track_Info_Gc_Value_Rec.Font_Id := Fonts.X_Load_Font
                                   (Track_Info_Display,"8x13bold");
Track_Info_Display_Context := Graphic_Output.X_Create_Gc
                  (Track_Info_Display, Track_Info_Drawing_Space,
```

```
                        Track_Info_Gc_Value_Mask,Track_Info_Gc_Value_Rec);
   X_Lib.X_Map_Window (Track_Info_Display, Track_Info_Window);
   X_Lib.Events.X_Flush (Track_Info_Display);
   X_Lib.Graphic_Output.X_Set_Background (Track_Info_Display,
                                          Track_Info_Display_Context, 7);
   X_Lib.Graphic_Output.X_Set_Foreground (Track_Info_Display,
                                          Track_Info_Display_Context, 1);

exception

   when Display_Not_Open  =>
      Text_Io.Put_Line ("Could not open Display");
     end Open_Track_Info_Display;
-------------------------------------------------------------------------
-   procedure Open_Alerts_Display is
     begin
       Alerts_Display := X_Lib.X_Open_Display ("");
      if (Alerts_Display = X_Lib.Null_Display) then
         raise Display_Not_Open;
      end if;
   Alerts_Main_Screen := X_Lib.Default_Screen (Alerts_Display);
   Alerts_Root_Wind := Root_Window (Alerts_Display, Alerts_Main_Screen);
   Alerts_Main_Attrs.Current_Cursor   := Cursors.X_Create_Font_Cursor
                                (Alerts_Display, X_Lib.Cursors.Xc_Man);
   Alerts_Main_Attrs.Border_Pixel     := 12;
   Alerts_Main_Attrs.Background_Pixel := 4;
   Alerts_Main_Attrs.Bit_Gravity      := Center_Gravity;
   Alerts_Main_Attrs.Window_Gravity   := Center_Gravity;
   Alerts_Main_Attrs.Event_Mask       := Events.Structure_Change_Mask;
   Alerts_Window :=
      X_Lib.X_Create_Window
```

```
            (Alerts_Display,
             Alerts_Root_Wind,
             Alerts_Window_X,
             Alerts_Window_Y,
             Alerts_Window_Width,
             Alerts_Window_Height,
             Alerts_Border_Width,
             X_Lib.Default_Depth (Alerts_Display, Alerts_Main_Screen),
             X_Lib.Input_Output,
             X_Lib.Default_Visual (Alerts_Display, Alerts_Main_Screen),
             Alerts_Wa_Values,
             Alerts_Main_Attrs);

      Events.X_Select_Input (Alerts_Display, Alerts_Window,
     X_Set_Icon_Name (Alerts_Display, Alerts_Window, "Alerts");
     X_Store_Name (Alerts_Display, Alerts_Window, "Alerts");
     X_Lib.Cursors.X_Define_Cursor (Alerts_Display, Alerts_Window,
                                    Alerts_Main_Attrs.Current_Cursor);
     Alerts_Drawing_Space := Drawable (Alerts_Window);
     Alerts_Gc_Value_Rec.Font_Id := Fonts.X_Load_Font
                                          (Alerts_Display, "8x13bold");
   Alerts_Display_Context := Graphic_Output.X_Create_Gc (Alerts_Display,
                             Alerts_Drawing_Space, Alerts_Gc_Value_Mask,
                             Alerts_Gc_Value_Rec);
   X_Lib.X_Map_Window (Alerts_Display, Alerts_Window);
   X_Lib.Events.X_Flush (Alerts_Display);
   X_Lib.Graphic_Output.X_Set_Background (Alerts_Display,
                                          Alerts_Display_Context, 12);
   X_Lib.Graphic_Output.X_Set_Foreground (Alerts_Display,
                                          Alerts_Display_Context, 1);
exception
```

```
   when Display_Not_Open  =>
      Text_Io.Put_Line ("Could not open Display");
end Open_Alerts_Display;
-------------------------------------------------------------------------

   procedure Open_Recommendations_Display is
     begin
   Recommendations_Display := X_Lib.X_Open_Display ("");
      if (Recommendations_Display = X_Lib.Null_Display) then
         raise Display_Not_Open;
      end if;
   Recommendations_Main_Screen := X_Lib.Default_Screen
                                          (Recommendations_Display);
   Recommendations_Root_Wind := X_Lib.Root_Window
                                       (Recommendations_Display,
                                        Recommendations_Main_Screen);
   Recommendations_Main_Attrs.Current_Cursor :=
                     Cursors.X_Create_Font_Cursor
                    (Recommendations_Display, X_Lib.Cursors.Xc_Gobbler);
   Recommendations_Main_Attrs.Border_Pixel     := 12;
   Recommendations_Main_Attrs.Background_Pixel := 6;
   Recommendations_Main_Attrs.Bit_Gravity      := Center_Gravity;
   Recommendations_Main_Attrs.Window_Gravity   := Center_Gravity;
   Recommendations_Main_Attrs.Event_Mask       :=
                                       Events.Structure_Change_Mask;
   Recommendations_Window :=
      X_Lib.X_Create_Window
          (Recommendations_Display,
           Recommendations_Root_Wind,
           Recommendations_Window_X,
```

```
        Recommendations_Window_Y,
        Recommendations_Window_Width,
        Recommendations_Window_Height,
        Recommendations_Border_Width,
        X_Lib.Default_Depth (Recommendations_Display,
                             Recommendations_Main_Screen),
        X_Lib.Input_Output,
        X_Lib.Default_Visual (Recommendations_Display,
                              Recommendations_Main_Screen),
        Recommendations_Wa_Values,
        Recommendations_Main_Attrs);

  Events.X_Select_Input (Recommendations_Display,
                         Recommendations_Window,
                         Recommendations_Main_Attrs.Event_Mask);
X_Set_Icon_Name (Recommendations_Display, Recommendations_Window,
                                          "Recommendations");
X_Store_Name (Recommendations_Display, Recommendations_Window,
                                       "Recommendations");
X_Lib.Cursors.X_Define_Cursor (Recommendations_Display,
                         Recommendations_Window,
                         Recommendations_Main_Attrs.Current_Cursor);
Recommendations_Drawing_Space := Drawable (Recommendations_Window);
Recommendations_Gc_Value_Rec.Font_Id :=
           Fonts.X_Load_Font (Recommendations_Display, "8x13bold");
Recommendations_Display_Context :=
    Graphic_Output.X_Create_Gc (Recommendations_Display,
                                Recommendations_Drawing_Space,
                                Recommendations_Gc_Value_Mask,
                                Recommendations_Gc_Value_Rec);
X_Lib.X_Map_Window (Recommendations_Display, Recommendations_Window);
```

```
   X_Lib.Events.X_Flush (Recommendations_Display);

   X_Lib.Graphic_Output.X_Set_Background (Recommendations_Display,
                                Recommendations_Display_Context, 6);
   X_Lib.Graphic_Output.X_Set_Foreground (Recommendations_Display,
                                Recommendations_Display_Context, 0);

exception

   when Display_Not_Open  =>
      Text_Io.Put_Line ("Could not open Display");
end Open_Recommendations_Display;

-------------------------------------------------------------------------

 procedure Open_Intel_Display is
     begin
   Intel_Display := X_Lib.X_Open_Display ("");
      if (Intel_Display = X_Lib.Null_Display) then
         raise Display_Not_Open;
      end if;
   Intel_Main_Screen := X_Lib.Default_Screen (Intel_Display);
   Intel_Root_Wind := Root_Window (Intel_Display, Intel_Main_Screen);
   Intel_Main_Attrs.Current_Cursor   :=
                     X_Lib.Cursors.X_Create_Font_Cursor
                    (Intel_Display, X_Lib.Cursors.Xc_Question_Arrow);
   Intel_Main_Attrs.Border_Pixel     := 12;
   Intel_Main_Attrs.Background_Pixel := 13;
   Intel_Main_Attrs.Bit_Gravity      := Center_Gravity;
   Intel_Main_Attrs.Window_Gravity   := Center_Gravity;
   Intel_Main_Attrs.Event_Mask       := Events.Structure_Change_Mask;
```

```
Intel_Window :=
     X_Lib.X_Create_Window
          (Intel_Display,
           Intel_Root_Wind,
           Intel_Window_X,
           Intel_Window_Y,
           Intel_Window_Width,
           Intel_Window_Height,
           Intel_Border_Width,
           X_Lib.Default_Depth (Intel_Display, Intel_Main_Screen),
           X_Lib.Input_Output,
           X_Lib.Default_Visual (Intel_Display, Intel_Main_Screen),
           Intel_Wa_Values,
           Intel_Main_Attrs);
    Events.X_Select_Input (Intel_Display, Intel_Window,
                           Intel_Main_Attrs.Event_Mask);
  X_Set_Icon_Name (Intel_Display, Intel_Window, "Intel");
  X_Store_Name (Intel_Display, Intel_Window, "Intelligence");
  X_Lib.Cursors.X_Define_Cursor (Intel_Display, Intel_Window,
                                 Intel_Main_Attrs.Current_Cursor);
  Intel_Drawing_Space := Drawable (Intel_Window);
  Intel_Gc_Value_Rec.Font_Id := Fonts.X_Load_Font
                                        (Intel_Display, "8x13bold");
  Intel_Display_Context := Graphic_Output.X_Create_Gc (Intel_Display,
                           Intel_Drawing_Space, Intel_Gc_Value_Mask,
                           Intel_Gc_Value_Rec);
  X_Lib.X_Map_Window (Intel_Display, Intel_Window);
  X_Lib.Events.X_Flush (Intel_Display);
  Graphic_Output.X_Set_Background
                          (Intel_Display, Intel_Display_Context, 13);
```

```
    X_Lib.Graphic_Output.X_Set_Foreground
                              (Intel_Display, Intel_Display_Context, 1);
exception
   when Display_Not_Open  =>
      Text_Io.Put_Line ("Could not open Display");
end Open_Intel_Display;


 procedure Open_System_Status_Display is
     begin
    System_Status_Display := X_Lib.X_Open_Display ("");
      if (System_Status_Display = X_Lib.Null_Display) then
         raise Display_Not_Open;
      end if;
   System_Status_Main_Screen := Default_Screen (System_Status_Display);
   System_Status_Root_Wind := X_Lib.Root_Window (System_Status_Display,
   System_Status_Main_Attrs.Current_Cursor   :=
                     X_Lib.Cursors.X_Create_Font_Cursor
                    (System_Status_Display, X_Lib.Cursors.Xc_Bogosity);
   System_Status_Main_Attrs.Border_Pixel     := 12;
   System_Status_Main_Attrs.Background_Pixel := 14;
   System_Status_Main_Attrs.Bit_Gravity      := Center_Gravity;
   System_Status_Main_Attrs.Window_Gravity   := Center_Gravity;
   System_Status_Main_Attrs.Event_Mask       :=
                                          Events.Structure_Change_Mask or
                                          Events.Exposure_Mask or
                                          Events.Button_Press_Mask;
   System_Status_Window :=
      X_Lib.X_Create_Window
          (System_Status_Display,
```

```
            System_Status_Root_Wind,
            System_Status_Window_X,
            System_Status_Window_Y,
            System_Status_Window_Width,
            System_Status_Window_Height,
            System_Status_Border_Width,
            Default_Depth
                    (System_Status_Display, System_Status_Main_Screen),
            X_Lib.Input_Output,
            X_Lib.Default_Visual
                    (System_Status_Display, System_Status_Main_Screen),
                    System_Status_Wa_Values, System_Status_Main_Attrs);
    Events.X_Select_Input (System_Status_Display, System_Status_Window,
                           System_Status_Main_Attrs.Event_Mask);
X_Set_Icon_Name (System_Status_Display, System_Status_Window,
                                        "System_Status");
X_Store_Name (System_Status_Display, System_Status_Window,
                                        "System Status");
X_Lib.Cursors.X_Define_Cursor (System_Status_Display,
                               System_Status_Window,
                              System_Status_Main_Attrs.Current_Cursor);
System_Status_Drawing_Space := Drawable (System_Status_Window);
System_Status_Gc_Value_Rec.Font_Id :=
                  Fonts.X_Load_Font (System_Status_Display,"8x13bold");
System_Status_Display_Context := Graphic_Output.X_Create_Gc
                                  System_Status_Display,
                                  System_Status_Drawing_Space,
                                  System_Status_Gc_Value_Mask,
                                  System_Status_Gc_Value_Rec);
X_Lib.X_Map_Window (System_Status_Display, System_Status_Window);
X_Lib.Events.X_Flush (System_Status_Display);
```

```
   X_Lib.Graphic_Output.X_Set_Background
            (System_Status_Display, System_Status_Display_Context, 14);
   X_Lib.Graphic_Output.X_Set_Foreground
            (System_Status_Display, System_Status_Display_Context, 1);
exception
   when Display_Not_Open  =>
      Text_Io.Put_Line ("Could not open Display");
end Open_System_Status_Display;
-------------------------------------------------------------------------
end Open_Interface_Displays;
```

APPENDIX B

MONITOR DISPLAYS

```
-------------------------------------------------------------------------
--       Author: Michael G. Stockwell
--       Date  : 12 September 1991
.........................................................................
--       Description:  Timers and Update Tasks for all displays.
-------------------------------------------------------------------------
with X_lib, Integration_System_Pkg, Tacplot_Pkg,
            Open_Interface_Displays, Process_Interface_Displays,
            Draw_Display_Graphics, System_Status_Pkg;
use  X_lib, Integration_System_Pkg, Tacplot_Pkg,
            Open_Interface_Displays, Process_Interface_Displays,
            Draw_Display_Graphics, System_Status_Pkg;


package Monitor_Displays is

   procedure Test_System_Status;
   task Monitor_Update_Intervals;

end Monitor_Displays;
```

```
package body Monitor_Displays is
  use Graphic_Output;
   procedure Test_System_Status is
     Eqpt_Status              : Status := DOWN;
     begin
       X_Clear_Window (System_Status_Display, System_Status_Window);
       X_Draw_Image_String
                  (System_Status_Display, System_Status_Drawing_Space,
                   System_Status_Display_Context,
                   5, 15,
                   "GPS System: ");
      INTEGRATION_SYSTEM.GET_SENSOR_STATUS ( GPS, Eqpt_Status );
        if Eqpt_Status  = DOWN then
          Write_Gps_Status_Down;
        else
          Write_Gps_Status_Up;
        end if;
      X_Draw_Image_String
                  (System_Status_Display, System_Status_Drawing_Space,
                   System_Status_Display_Context,
                   5, 45,
                   "Link 11    : ");
      INTEGRATION_SYSTEM.GET_SENSOR_STATUS ( LINK, Eqpt_Status );
        if Eqpt_Status  = DOWN then
          Write_Link_Status_Down;
        else
          Write_Link_Status_Up;
        end if;
```

```
X_Draw_Image_String
            (System_Status_Display, System_Status_Drawing_Space,
            System_Status_Display_Context,
            5, 30,
            "Radar 1    : ");
INTEGRATION_SYSTEM.GET_SENSOR_STATUS ( Radar, Eqpt_Status );
  if Eqpt_Status  = DOWN then
    Write_Radar_Status_Down;
  else
    Write_Radar_Status_Up;
  end if;
X_Draw_Image_String
            (System_Status_Display, System_Status_Drawing_Space,
             System_Status_Display_Context,
             250, 15,
             "Fathometer: ");
INTEGRATION_SYSTEM.GET_SENSOR_STATUS ( Fathometer, Eqpt_Status );
  if Eqpt_Status  = DOWN then
    Write_Fath_Status_Down;
  else
    Write_Fath_Status_Up;
  end if;
X_Draw_Image_String
            (System_Status_Display, System_Status_Drawing_Space,
             System_Status_Display_Context,
             250, 30,
             "Gyro       : ");
INTEGRATION_SYSTEM.GET_SENSOR_STATUS ( Gyro, Eqpt_Status );
  if Eqpt_Status  = DOWN then
    Write_Gyro_Status_Down;
  else
```

```
        Write_Gyro_Status_Up;
      end if;
    X_Draw_Image_String
                (System_Status_Display, System_Status_Drawing_Space,
                 System_Status_Display_Context,
                 250, 45,
                 "Pitsword  : ");
    INTEGRATION_SYSTEM.GET_SENSOR_STATUS ( Pitsword, Eqpt_Status );
      if Eqpt_Status  = DOWN then
        Write_Sword_Status_Down;
      else
        Write_Sword_Status_Up;
      end if;
end Test_System_Status;

task body Monitor_Update_Intervals is
SECONDS                    : constant DURATION := 1.0;
INTERVAL                   : constant DURATION := 4 * SECONDS;

  begin
    delay (Interval);
      loop
        begin
          delay (Interval);
          INTEGRATION_SYSTEM.FILL_TACPLOT;
          DISPLAY_TACPLOT;
          exception
          when others => PUT_LINE ("Exception in Timer Task");
        end;
      end loop;
  end Monitor_Update_Intervals;
```

```
begin
   null;
end Monitor_Displays;
```

APPENDIX C

PROCESS INTERFACE DISPLAYS

```
--        Author: Michael G. Stockwell
--        Date  : 21 September 1991
--        Description: Handles all information processing prior to writing
                       that information to the screen.

with Track_Pkg, Cpa_Pkg, Absolute_Time_Pkg, Global_Observation_Pkg,
X_Lib, VStrings, Calendar, System_Status_Pkg, Draw_Display_Graphics,
Vector_2_Pkg, Text_IO, Angle_Pkg, Speed_Pkg, Velocity_Pkg,
Global_Position_Pkg, Relative_Position_Pkg, Relative_Time_Pkg,
Distance_Pkg;

use  Track_Pkg, Cpa_Pkg, Absolute_Time_Pkg, Global_Observation_Pkg,
X_Lib, Velocity_Pkg, Draw_Display_Graphics, Vector_2_Pkg, Text_IO,
Angle_Pkg, Speed_Pkg, Calendar, System_Status_Pkg, Global_Position_Pkg,
Relative_Position_Pkg, Relative_Time_Pkg;

package Process_Interface_Displays is

package APKG renames TRACK_PKG.AMP_STR;
use APKG;
package VPKG renames TRACK_PKG.V_AND_C_STR;
use VPKG;
```

```
package Test_Float_IO is new Float_IO (Float);
package Test_Integer_IO is new Integer_IO (Integer);
use Test_Float_IO, Test_Integer_IO;

 procedure Process_Tacplot_Objects (Current_Track:   in Track);
 procedure Process_Hook_Position  (Hook_Position_X: in X_lib.Coordinate;
                                   Hook_Position_Y: in X_lib.Coordinate);
 procedure Set_AV_Alerts;
 procedure Set_Visual_Alerts;
 procedure Disable_Alerts;
 procedure Set_CPA_Alert_Range (Alert_Range: in Float);
 procedure Set_Filter_Status   (Stat       : in String);

 function Get_Ownship_Track return Track;
 function Get_Hooked_Track_Number return Natural;
 function Get_Hooked_Track_Object return Track;
--------------------------------------------------------------------------
package body Process_Interface_Displays is
subtype Coord is X_Lib.Coordinate;

Ownship_Track                :       Track;
Ownship_Name                 :       String (1..12) := "USS Spanagel";
Hooked_Track_Flag            :       Boolean := False;
No_Alert_Flag                :       Boolean := False;
Vis_Alert_Flag               :       Boolean := False;
Filter_Set_Flag              :       Boolean := False;
Nav_Man_Flag                 :       Boolean := True;
CPA_Alert_Range              :       Float := 3000.0;
Hooked_Track                 :       Track;
Hooked_Track_No              :       Natural;
Hook_X                       :       Coord := 0;
```

```
Hook_Y                          :       Coord := 0;
Screen_Scale_Factor             :       Float := 32000.0;
Target_Position_Scale_Factor    :       Float := 91.43;
Surface_Velocity_Scale_Factor   :       Coord := 3;
Air_Velocity_Scale_Factor       :       Coord := 10;
Scale_Selector_Up_Center_X      :       Coord := 65;
Scale_Selector_Up_Center_Y      :       Coord := 40;
Scale_Selector_Down_Center_X    :       Coord := 25;
Scale_Selector_Down_Center_Y    :       Coord := 40;

  procedure Process_Tacplot_Objects (Current_Track:   in Track) is
    Current_Track_Category              :       Track_Category;
    Point_Category                      :       Special_Point_Category;
    CBDR_Alert_Range                    :       Float := 500.0;
    Target_Identity                     :       Identity_Type;
    Target_Position_X                   :       Coord;
    Target_Position_Y                   :       Coord;
    Target_Coordinate_X                 :       Float;
    Target_Coordinate_Y                 :       Float;
    Target_Relative_Position            :       Relative_Position;
    Ownship_Relative_Position           :       Relative_Position;
    Velocity_X                          :       Coord;
    Velocity_Y                          :       Coord;
    Time_Now                            :       Absolute_Time;
    VES                                 :       String (1..80);
    AMP                                 :       String (1..80);
    Track_No                            :       Natural;
    Tgt_Cpa                             :       Cpa_Type;
    Tgt_Cpa_Range                       :       Float;
    Tgt_Cpa_Bearing                     :       Float;
    Tgt_Cpa_Hours                       :       Natural;
```

```
    Tgt_Cpa_Mins                    :       Natural;
    Tgt_Lat_Deg                     :       Natural;
    Tgt_Lat_Min                     :       Natural;
    Tgt_Lat_Sec                     :       Natural;
    Tgt_Lat_Dir                     :       North_South;
    Tgt_Long_Deg                    :       Natural;
    Tgt_Long_Min                    :       Natural;
    Tgt_Long_Sec                    :       Natural;
    Tgt_Long_Dir                    :       East_West;

    function Compute_Surface_Velocity_X ( Current_Track: in Track)
return Coord is
      begin
       return
         ((Coord(Integer(X_Coordinate(True_Velocity(Current_Track)))))
                                    * Surface_Velocity_Scale_Factor);
      end Compute_Surface_Velocity_X;
    function Compute_Air_Velocity_X ( Current_Track: in Track)
    return Coord is
      begin
       return
         ((Coord(Integer(X_Coordinate(True_Velocity(Current_Track)))))
                                      / Air_Velocity_Scale_Factor);
      end Compute_Air_Velocity_X;
    function Compute_Surface_Velocity_Y ( Current_Track: in Track)
    return Coord is
      begin
       return
        ((Coord(Integer(Y_Coordinate(True_Velocity(Current_Track)))))
                                    * Surface_Velocity_Scale_Factor);
      end Compute_Surface_Velocity_Y;
```

```
  function Compute_Air_Velocity_Y ( Current_Track: in Track)
  return Coord is
    begin
     return
       ((Coord(Integer(Y_Coordinate(True_Velocity(Current_Track)))))
                                          / Air_Velocity_Scale_Factor);
    end Compute_Air_Velocity_Y;

begin
  Time_NOW := ABSOLUTE_TIME_PKG.NOW;
  if Track_ID_Number(Current_Track) = 0 then
    Clear_Interface_Displays;
    Draw_Display_Attributes;

    if Nav_Man_Flag then
     Write_Nav_Status ("Manual");
    else
      Write_Nav_Status ("Automatic");
    end if;

    if Filter_Set_Flag then
     Write_Filters_Status ("Set");
    else
      Write_Filters_Status ("None");
    end if;

    if No_Alert_Flag then
     Write_Alerts_Status ("None");
    elsif Vis_Alert_Flag then
     Write_Alerts_Status ("Visual");
    else
```

```
 Write_Alerts_Status ("Both");
end if;
Write_Range_Scale
           (Natural'Image(Natural(Screen_Scale_Factor/2000.0)));
Write_Hours  (Natural'Image(Hours(Time_of_Day(Time_NOW))));
Write_Mins  (Natural'Image(Minutes(Time_of_Day(Time_NOW))));
Ownship_Track := Current_Track;
Velocity_X := Compute_Surface_Velocity_X (Current_Track);
Velocity_Y := Compute_Surface_Velocity_Y (Current_Track);
Draw_Tacplot_Speed_Leader (Centerpoint_X, Centerpoint_Y,
                           Velocity_X, Velocity_Y);

if Hooked_Track_Flag and (Hooked_Track_No = 0) then
 Draw_Tacplot_Hook (Centerpoint_X, Centerpoint_Y);
 Write_Tgt_Cse  (Natural'Image(Natural(Radians_to_Degrees
              (True_Course(Ownship_Track)))));
 Write_Tgt_Spd  (Natural'Image(Natural(Speed_in_Knots
              (True_Speed(Ownship_Track)))));
 Write_Tgt_Id   ("FRIENDLY");
 Write_Tgt_Update_Hours  (Natural'Image(Hours
                         (Time_of_Day(Most_Recent_Observation
                         (Ownship_Track).Observation_Time))));
 Write_Tgt_Update_Mins   (Natural'Image(Minutes(Time_of_Day
                         (Most_Recent_Observation
                         (Ownship_Track).Observation_Time))));
 Write_Tgt_Class   ("OWNSHIP");
 Write_Tgt_Name    ("USS SPANAGEL");

 Get_Latitude  ((Current_Position(Ownship_Track)),
             Tgt_Lat_Dir, Tgt_Lat_Deg, Tgt_Lat_Min, Tgt_Lat_Sec);
 Get_Longitude ((Current_Position(Ownship_Track)),
```

```
                    Tgt_Long_Dir, Tgt_Long_Deg, Tgt_Long_Min,
                    Tgt_Long_Sec);
    Write_Tgt_Lat_Position (Natural'Image(Tgt_Lat_Deg),
                           Natural'Image(Tgt_Lat_Min),
                           Natural'Image(Tgt_Lat_Sec),
                           North_South'Image(Tgt_Lat_Dir));
    Write_Tgt_Long_Position(Natural'Image(Tgt_Long_Deg),
                           Natural'Image(Tgt_Long_Min),
                           Natural'Image(Tgt_Long_Sec),
                           East_West'Image(Tgt_Long_Dir));
  end if;
  return;
else
  Ownship_Relative_Position := Find_Relative_Position
            ((Current_Position(Ownship_Track)),
             (Current_Position(Current_Track)));

  Target_Relative_Position := Find_Relative_Position
            ((Current_Position(Current_Track)),
             (Current_Position(Ownship_Track)));

  if (RANGE_OF ( Target_Relative_Position))
      > Screen_Scale_Factor then
    return;
  end if;
end if;

Track_No         := Natural(Track_Id_Number(Current_Track));
Tgt_Cpa          := Find_CPA (Current_Track, Ownship_Track);
Tgt_Cpa_Range    := Length(Tgt_CPA.CPA_Bearing_and_Range);
Tgt_Cpa_Bearing := Radians_to_Degrees
```

```
                    (Direction(Tgt_CPA.CPA_Bearing_and_Range));
Tgt_Cpa_Hours    := Hours(Time_of_Day(Tgt_CPA.Time_of_CPA));
Tgt_Cpa_Mins     := Minutes(Time_of_Day(Tgt_CPA.Time_of_CPA));

if not No_Alert_Flag and (Tgt_Cpa_Range < CPA_Alert_Range) then

  Write_CPA_Alert ((Natural'Image(Track_No)),
                   (Natural'Image(Natural(Tgt_Cpa_Bearing))),
                   (Natural'Image(Natural(Tgt_Cpa_Range))),
                   (Natural'Image(Tgt_Cpa_Hours)),
                   (Natural'Image(Tgt_Cpa_Mins)));
   if not Vis_Alert_Flag then
     put (character'val(7));
   end if;
 end if;

Target_Coordinate_X := (X_Coordinate(Target_Relative_Position))
                     / Target_Position_Scale_Factor;
Target_Coordinate_Y := (Y_Coordinate(Target_Relative_Position))
                    / Target_Position_Scale_Factor;
Target_Position_X := Centerpoint_X +
                              (Coord(Integer(Target_Coordinate_X)));
Target_Position_Y := Centerpoint_Y -
                              (Coord(Integer(Target_Coordinate_Y)));

if (Hooked_Track_Flag = True) and
   (Hooked_Track_No = Natural(Track_Id_Number(Current_Track))) then

  Draw_Tacplot_Hook (Target_Position_X, Target_Position_Y);
  Write_Tgt_Cse (Natural'Image(Natural(Radians_to_Degrees
              (True_Course(Current_Track)))));
```

```
Write_Tgt_Spd (Natural'Image(Natural(Speed_in_Knots
            (True_Speed(Current_Track)))));
Write_Tgt_Brg
          (Natural'Image(Natural(Radians_to_Degrees(True_Bearing
          (Ownship_Track, Current_Track)))));
Write_Tgt_Range (Natural'Image(Natural
                (Range_of(Target_Relative_Position))));
Write_Tgt_No
       (Natural'Image(Natural(Track_Id_Number(Current_Track))));
Write_Tgt_Ctl_Type  (Control_Type'Image(Control(Current_Track)));
Write_Tgt_Update_Hours (Natural'Image(Hours
                       (Time_of_Day(Most_Recent_Observation
                       (Current_Track).Observation_Time))));
Write_Tgt_Update_Mins  (Natural'Image(Minutes
                       (Time_of_Day(Most_Recent_Observation
                       (Current_Track).Observation_Time))));
Get_Latitude  ((Current_Position(Current_Track)),
             Tgt_Lat_Dir, Tgt_Lat_Deg, Tgt_Lat_Min, Tgt_Lat_Sec);
Get_Longitude ((Current_Position(Current_Track)),
              Tgt_Long_Dir, Tgt_Long_Deg, Tgt_Long_Min,
              Tgt_Long_Sec);
Write_Tgt_Lat_Position (Natural'Image(Tgt_Lat_Deg),
                       Natural'Image(Tgt_Lat_Min),
                       Natural'Image(Tgt_Lat_Sec),
                       North_South'Image(Tgt_Lat_Dir));
Write_Tgt_Long_Position(Natural'Image(Tgt_Long_Deg),
                       Natural'Image(Tgt_Long_Min),
                       Natural'Image(Tgt_Long_Sec),
                       East_West'Image(Tgt_Long_Dir));
Write_Rel_Brg (Natural'Image(Natural(Radians_to_Degrees
            (Relative_Bearing(Ownship_Track, Current_Track)))));
```

```
Write_Rel_Spd (Natural'Image(Natural(Speed_In_Knots(Spd
    (Target_Relative_Velocity(Ownship_Track, Current_Track))))));
Write_Tgt_Angle (Natural'Image(Natural(Radians_to_Degrees
            (Relative_Bearing(Current_Track,Ownship_Track)))));
Write_Tgt_Amp_Info (APKG.STR(Ampl_Info(Current_Track)));

if ((Trk_Category(Current_Track)) = Surface_Platform) or
   ((Trk_Category(Current_Track)) = Subsurface_Platform) then
   Write_Tgt_Class (VPKG.STR(Platform_Class(Current_Track)));
   Write_Tgt_Id
            (Identity_Type'Image(Track_Identity(Current_Track)));
   Write_Tgt_Name (VPKG.STR(Vessel_Name(Current_Track)));
end if;

if ((Trk_Category(Current_Track)) = Air_Platform) then
   Write_Tgt_Class (VPKG.STR(Platform_Class(Current_Track)));
   Write_Tgt_Id
            (Identity_Type'Image(Track_Identity(Current_Track)));
   Write_Tgt_Height (Integer'Image(Integer
                  ((Altitude(Current_Track)* 3.0))));
end if;

if  (Tgt_CPA.Time_of_CPA)  <  ABSOLUTE_TIME_PKG.NOW then
  Write_Tgt_CPA_PNO;
elsif Tgt_Cpa_Range < 500.0 then
 Write_Tgt_CPA_CBDR;
else
 Write_Tgt_CPA_Bearing (Natural'Image(Natural(Tgt_Cpa_Bearing)));
 Write_Tgt_CPA_Range    (Natural'Image(Natural(Tgt_Cpa_Range)));
 Write_Tgt_CPA_Hours    (Natural'Image(Tgt_Cpa_Hours));
 Write_Tgt_CPA_Mins     (Natural'Image(Tgt_Cpa_Mins));
```

```
    end if;
  end if;
  if Hooked_Track_Flag = False and
                          (abs(Hook_X - Target_Position_X)) < 10 and
                          (abs(Hook_Y - Target_Position_Y)) < 10 then
    Hooked_Track_Flag := True;
    Hooked_Track := Current_Track;
    Hooked_Track_No := Natural(Track_Id_Number(Current_Track));
    Draw_Tacplot_Hook (Target_Position_X, Target_Position_Y);
    Write_Tgt_Cse  (Natural'Image(Natural(Radians_to_Degrees
                   (True_Course(Current_Track)))));
    Write_Tgt_Spd  (Natural'Image(Natural(Speed_in_Knots
                   (True_Speed(Current_Track)))));
    Write_Tgt_Brg
              (Natural'Image(Natural(Radians_to_Degrees(True_Bearing
              (Ownship_Track, Current_Track)))));
    Write_Tgt_Range  (Natural'Image(Natural
                     (Range_of(Target_Relative_Position))));
    Write_Tgt_No
           (Natural'Image(Natural(Track_Id_Number(Current_Track))));
    Write_Tgt_Ctl_Type   (Control_Type'Image(Control(Current_Track)));
    Write_Tgt_Update_Hours  (Natural'Image(Hours
                            (Time_of_Day(Most_Recent_Observation
                            (Current_Track).Observation_Time))));
    Write_Tgt_Update_Mins   (Natural'Image(Minutes(Time_of_Day
                         (Most_Recent_Observation
                         (Current_Track).Observation_Time))));
    Get_Latitude   ((Current_Position(Current_Track)),
                  Tgt_Lat_Dir, Tgt_Lat_Deg, Tgt_Lat_Min, Tgt_Lat_Sec);
    Get_Longitude  ((Current_Position(Current_Track)),
                   Tgt_Long_Dir, Tgt_Long_Deg, Tgt_Long_Min,
```

```
            Tgt_Long_Sec);
Write_Tgt_Lat_Position (Natural'Image(Tgt_Lat_Deg),
                        Natural'Image(Tgt_Lat_Min),
                        Natural'Image(Tgt_Lat_Sec),
                        North_South'Image(Tgt_Lat_Dir));
Write_Tgt_Long_Position(Natural'Image(Tgt_Long_Deg),
                        Natural'Image(Tgt_Long_Min),
                        Natural'Image(Tgt_Long_Sec),
                        East_West'Image(Tgt_Long_Dir));
Write_Rel_Brg (Natural'Image(Natural(Radians_to_Degrees
             (Relative_Bearing(Ownship_Track, Current_Track)))));
Write_Rel_Spd (Natural'Image(Natural(Speed_In_Knots(Spd
               (Target_Relative_Velocity(Ownship_Track,
                Current_Track))))));
Write_Tgt_Angle (Natural'Image(Natural(Radians_to_Degrees
               (Relative_Bearing(Current_Track,Ownship_Track)))));
Write_Tgt_Amp_Info (APKG.STR(Ampl_Info(Current_Track)));

if ((Trk_Category(Current_Track)) = Surface_Platform) or
   ((Trk_Category(Current_Track)) = Subsurface_Platform) then
   Write_Tgt_Class (VPKG.STR(Platform_Class(Current_Track)));
   Write_Tgt_Id
           (Identity_Type'Image(Track_Identity(Current_Track)));
   Write_Tgt_Name (VPKG.STR(Vessel_Name(Current_Track)));
end if;

if ((Trk_Category(Current_Track)) = Air_Platform) then
   Write_Tgt_Class (VPKG.STR(Platform_Class(Current_Track)));
   Write_Tgt_Id
           (Identity_Type'Image(Track_Identity(Current_Track)));
   Write_Tgt_Height (Integer'Image(Integer
```

```
                    ((Altitude(Current_Track)* 3.0))));
    end if;

    if  (Tgt_CPA.Time_of_CPA)  <  ABSOLUTE_TIME_PKG.NOW then
      Write_Tgt_CPA_PNO;
    elsif Tgt_Cpa_Range < 500.0 then
      Write_Tgt_CPA_CBDR;
    else
     Write_Tgt_CPA_Bearing (Natural'Image(Natural(Tgt_Cpa_Bearing)));
     Write_Tgt_CPA_Range   (Natural'Image(Natural(Tgt_Cpa_Range)));
     Write_Tgt_CPA_Hours   (Natural'Image(Tgt_Cpa_Hours));
     Write_Tgt_CPA_Mins    (Natural'Image(Tgt_Cpa_Mins));
    end if;
  end if;
    Current_Track_Category := Trk_Category(Current_Track);
     Case Current_Track_Category is
       when Unknown =>
         Velocity_X := Compute_Surface_Velocity_X (Current_Track);
         Velocity_Y := Compute_Surface_Velocity_Y (Current_Track);
         Draw_Tacplot_Unknown (Target_Position_X, Target_Position_Y);
         Draw_Tacplot_Speed_Leader
                              (Target_Position_X, Target_Position_Y,
                               Velocity_X, Velocity_Y);
         Draw_Tacplot_Track_No (Target_Position_X, Target_Position_Y,
                                Natural'Image(Track_No));
       when Surface_Platform =>
       Target_Identity := Track_Identity (Current_Track);
         Case Target_Identity is
           when Friendly =>
            Velocity_X := Compute_Surface_Velocity_X (Current_Track);
            Velocity_Y := Compute_Surface_Velocity_Y (Current_Track);
```

```
            Draw_Tacplot_Friendly_Surface (Target_Position_X,
                                           Target_Position_Y);
            Draw_Tacplot_Speed_Leader
                          (Target_Position_X, Target_Position_Y,
                             Velocity_X, Velocity_Y);
            Draw_Tacplot_Track_No (Target_Position_X,
                                   Target_Position_Y,
                                   Natural'Image(Track_No));

         when Hostile =>
           Velocity_X := Compute_Surface_Velocity_X (Current_Track);
           Velocity_Y := Compute_Surface_Velocity_Y (Current_Track);
           Draw_Tacplot_Hostile_Surface (Target_Position_X,
                                         Target_Position_Y);
           Draw_Tacplot_Speed_Leader
                          (Target_Position_X, Target_Position_Y,
                           Velocity_X, Velocity_Y);
           Draw_Tacplot_Track_No (Target_Position_X,
                                  Target_Position_Y,
                                  Natural'Image(Track_No));
         when Neutral =>
           Velocity_X := Compute_Surface_Velocity_X (Current_Track);
           Velocity_Y := Compute_Surface_Velocity_Y (Current_Track);
           Draw_Tacplot_Neutral_Surface (Target_Position_X,
                                         Target_Position_Y);
           Draw_Tacplot_Speed_Leader
                          (Target_Position_X, Target_Position_Y,
                           Velocity_X, Velocity_Y);
           Draw_Tacplot_Track_No (Target_Position_X,
                                  Target_Position_Y,
                                  Natural'Image(Track_No));
```

```
when Unknown =>
   Velocity_X := Compute_Surface_Velocity_X (Current_Track);
   Velocity_Y := Compute_Surface_Velocity_Y (Current_Track);
   Draw_Tacplot_Unknown_Surface (Target_Position_X,
                                 Target_Position_Y);
   Draw_Tacplot_Speed_Leader
                    (Target_Position_X, Target_Position_Y,
                     Velocity_X, Velocity_Y);
   Draw_Tacplot_Track_No (Target_Position_X,
                          Target_Position_Y,
                          Natural'Image(Track_No));
    when others => null;
     end case;
when Subsurface_Platform =>
Target_Identity := Track_Identity (Current_Track);

Case Target_Identity is
  when Friendly =>
   Velocity_X := Compute_Surface_Velocity_X (Current_Track);
   Velocity_Y := Compute_Surface_Velocity_Y (Current_Track);
   Draw_Tacplot_Friendly_SubSurface (Target_Position_X,
                                     Target_Position_Y);
   Draw_Tacplot_Speed_Leader
                (Target_Position_X, Target_Position_Y,
                 Velocity_X, Velocity_Y);
   Draw_Tacplot_Track_No (Target_Position_X,
                          Target_Position_Y,
                          Natural'Image(Track_No));
```

```
when Hostile =>
   Velocity_X := Compute_Surface_Velocity_X (Current_Track);
   Velocity_Y := Compute_Surface_Velocity_Y (Current_Track);
   Draw_Tacplot_Hostile_SubSurface (Target_Position_X,
                                    Target_Position_Y);
   Draw_Tacplot_Speed_Leader
                    (Target_Position_X, Target_Position_Y,
                     Velocity_X, Velocity_Y);
   Draw_Tacplot_Track_No (Target_Position_X,
                          Target_Position_Y,
                          Natural'Image(Track_No));
 when Neutral =>
   Velocity_X := Compute_Surface_Velocity_X (Current_Track);
   Velocity_Y := Compute_Surface_Velocity_Y (Current_Track);
   Draw_Tacplot_Neutral_SubSurface (Target_Position_X,
                                    Target_Position_Y);
   Draw_Tacplot_Speed_Leader
                (Target_Position_X, Target_Position_Y,
                 Velocity_X, Velocity_Y);
   Draw_Tacplot_Track_No (Target_Position_X,
                          Target_Position_Y,
                          Natural'Image(Track_No));
when Unknown =>
   Velocity_X := Compute_Surface_Velocity_X (Current_Track);
   Velocity_Y := Compute_Surface_Velocity_Y (Current_Track);
   Draw_Tacplot_Unknown_SubSurface (Target_Position_X,
                                    Target_Position_Y);
   Draw_Tacplot_Speed_Leader
                  (Target_Position_X, Target_Position_Y,
                   Velocity_X, Velocity_Y);
    Draw_Tacplot_Track_No (Target_Position_X,
```

```
                                    Target_Position_Y,
                                    Natural'Image(Track_No));
        when others => null;
      end case;
  when Air_Platform =>
  Target_Identity := Track_Identity (Current_Track);
    Case Target_Identity is
      when Friendly =>
        Velocity_X := Compute_Air_Velocity_X (Current_Track);
        Velocity_Y := Compute_Air_Velocity_Y (Current_Track);
        Draw_Tacplot_Friendly_Air (Target_Position_X,
                                   Target_Position_Y);
        Draw_Tacplot_Speed_Leader
                        (Target_Position_X, Target_Position_Y,
                         Velocity_X, Velocity_Y);
        Draw_Tacplot_Track_No (Target_Position_X,
                               Target_Position_Y,
                               Natural'Image(Track_No));
      when Hostile =>
        Velocity_X := Compute_Air_Velocity_X (Current_Track);
        Velocity_Y := Compute_Air_Velocity_Y (Current_Track);
        Draw_Tacplot_Hostile_Air (Target_Position_X,
                                  Target_Position_Y);
        Draw_Tacplot_Speed_Leader
                       (Target_Position_X, Target_Position_Y,
                        Velocity_X, Velocity_Y);
        Draw_Tacplot_Track_No (Target_Position_X,
                               Target_Position_Y,
                               Natural'Image(Track_No));
```

```
    when Neutral =>
      Velocity_X := Compute_Air_Velocity_X (Current_Track);
      Velocity_Y := Compute_Air_Velocity_Y (Current_Track);
       Draw_Tacplot_Neutral_Air (Target_Position_X,
                                 Target_Position_Y);
       Draw_Tacplot_Speed_Leader
                        (Target_Position_X, Target_Position_Y,
                         Velocity_X, Velocity_Y);
       Draw_Tacplot_Track_No (Target_Position_X,
                              Target_Position_Y,
                              Natural'Image(Track_No));
    when Unknown =>
      Velocity_X := Compute_Air_Velocity_X (Current_Track);
      Velocity_Y := Compute_Air_Velocity_Y (Current_Track);
      Draw_Tacplot_Unknown_Air (Target_Position_X,
                                Target_Position_Y);
      Draw_Tacplot_Speed_Leader
                        (Target_Position_X, Target_Position_Y,
                         Velocity_X, Velocity_Y);
      Draw_Tacplot_Track_No (Target_Position_X,
                             Target_Position_Y,
                             Natural'Image(Track_No));
  when others => null;
end case;
when Region => null;
when Path => null;
when Special_Point =>
  Point_Category := Spec_Point_Category (Current_Track);
    case Point_Category is
    when General =>
    Velocity_X := Compute_Surface_Velocity_X (Current_Track);
```

```
      Velocity_Y := Compute_Surface_Velocity_Y (Current_Track);
       Draw_Tacplot_Ref_Point (Target_Position_X,
                               Target_Position_Y);
       Draw_Tacplot_Speed_Leader
                       (Target_Position_X, Target_Position_Y,
                        Velocity_X, Velocity_Y);
       Draw_Tacplot_Track_No (Target_Position_X,
                              Target_Position_Y,
                              Natural'Image(Track_No));
    when Waypoint =>
      Velocity_X := Compute_Surface_Velocity_X (Current_Track);
      Velocity_Y := Compute_Surface_Velocity_Y (Current_Track);
      Draw_Tacplot_WayPoint (Target_Position_X,
                             Target_Position_Y);
      Draw_Tacplot_Speed_Leader
             (Target_Position_X, Target_Position_Y,
              Velocity_X, Velocity_Y);
      Draw_Tacplot_Track_No (Target_Position_X,
                             Target_Position_Y,
                             Natural'Image(Track_No));
   when Nav_Hazard =>
     Velocity_X := Compute_Surface_Velocity_X (Current_Track);
     Velocity_Y := Compute_Surface_Velocity_Y (Current_Track);
     Draw_Tacplot_Nav_Hazard (Target_Position_X,
                              Target_Position_Y);
     Draw_Tacplot_Speed_Leader
                      (Target_Position_X, Target_Position_Y,
                       Velocity_X, Velocity_Y);
     Draw_Tacplot_Track_No (Target_Position_X,
                            Target_Position_Y,
                            Natural'Image(Track_No));
```

```
            when Others => null;
          end case;
        when Man_in_Water =>
          Velocity_X := Compute_Surface_Velocity_X (Current_Track);
          Velocity_Y := Compute_Surface_Velocity_Y (Current_Track);
          Draw_Tacplot_Speed_Leader
                        (Target_Position_X, Target_Position_Y,
                         Velocity_X, Velocity_Y);
          Draw_Tacplot_Man_in_Water (Target_Position_X,
                                     Target_Position_Y);
          Draw_Tacplot_Track_No (Target_Position_X, Target_Position_Y,
                                 Natural'Image(Track_No));
        when others => null;
      end case;
end Process_Tacplot_Objects;
procedure Process_Hook_Position(Hook_Position_X: in X_lib.Coordinate;
                                Hook_Position_Y: in X_lib.Coordinate) is
  begin
    if (abs(Hook_Position_X - Scale_Selector_Up_Center_X)) < 15 and
       (abs(Scale_Selector_Up_Center_Y - Hook_Position_Y)) < 15 and
       (Screen_Scale_Factor < (512.0 * 2000.0)) then
        Screen_Scale_Factor := (Screen_Scale_Factor * 2.0);
        Target_Position_Scale_Factor :=
                                (Target_Position_Scale_Factor * 2.0);
        Return;
     elsif
       (abs(Hook_Position_X - Scale_Selector_Down_Center_X)) < 15 and
        (abs(Scale_Selector_Down_Center_Y - Hook_Position_Y)) < 15 and
        (Screen_Scale_Factor > 2000.0) then
         Screen_Scale_Factor := (Screen_Scale_Factor / 2.0);
```

```
        Target_Position_Scale_Factor :=
                                    (Target_Position_Scale_Factor / 2.0);
         Return;
       else
        Hooked_Track_Flag := False;
        Hooked_Track_No := 9999;
        Hook_X := Hook_Position_X;
        Hook_Y := Hook_Position_Y;

         if (abs(Hook_Position_X - Centerpoint_X)) < 10 and
            (abs(Centerpoint_Y - Hook_Position_Y)) < 10 then
           Hooked_Track_Flag := True;
           Hooked_Track := Ownship_Track;
           Hooked_Track_No := 0;
         end if;
       end if;
end Process_Hook_Position;

procedure Set_AV_Alerts is
  begin
     Vis_Alert_Flag := False;
     No_Alert_Flag  := False;
  end Set_AV_Alerts;

procedure Set_Visual_Alerts is
  begin
    Vis_Alert_Flag := True;
    No_Alert_Flag  := False;
  end Set_Visual_Alerts;
```

```
procedure Disable_Alerts is
  begin
    No_Alert_Flag := True;
  end Disable_Alerts;

procedure Set_CPA_Alert_Range (Alert_Range: in Float) is
  begin
    CPA_Alert_Range := Alert_Range;
  end Set_CPA_Alert_Range;

procedure Set_Filter_Status (Stat: in String) is
  begin
     if Stat = "ON" then
       Filter_Set_Flag := True;
     else
       Filter_Set_Flag := False;
     end if;
  end Set_Filter_Status;

function Get_Ownship_Track return Track is
   begin
      return Ownship_Track;
   end Get_Ownship_Track;

function Get_Hooked_Track_Number return Natural is
  begin
    if Hooked_Track_Flag then
      return Track_ID_Number (Hooked_Track);
    else return 9999;
    end if;
  end Get_Hooked_Track_Number;
```

```
    function Get_Hooked_Track_Object return Track is
      begin
        return Hooked_Track;
      end Get_Hooked_Track_Object;

end Process_Interface_Displays;
```

APPENDIX D

DRAW DISPLAY GRAPHICS

```
--        Author: Michael G. Stockwell
--        Date  : 21 September 1991
--        Description: Handles all screen drawing and writing functions

with  X_Lib, Open_Interface_Displays;
use   X_Lib, Open_Interface_Displays;

package Draw_Display_Graphics is
   use Graphic_Output;
   Box_Width                : X_Lib.Pixel      := 20;
   Box_Height               : X_Lib.Pixel      := 20;
   Centerpoint_X            : X_Lib.Coordinate := 350;
   Centerpoint_Y            : X_Lib.Coordinate := 350;
   Hook_Width               : X_Lib.Pixel      :=40;
   Hook_Height              : X_Lib.Pixel      :=40;
   Neutral_Symbol           : String (1..1)    := "N";

-- Initializing Display Procedures

   procedure Clear_Interface_Displays;
   procedure Draw_Display_Attributes;
```

```
-- Tacplot General Symbol Drawing Procedures
   procedure Draw_Tacplot_Hook (Track_Location_X   : in Coordinate;
                                Track_Location_Y   : in Coordinate);

   procedure Draw_Tacplot_Speed_Leader
                               (Track_Location_X   : in Coordinate;
                                Track_Location_Y   : in Coordinate;
                                Speed_Leader_X     : in Coordinate;
                                Speed_Leader_Y     : in Coordinate);
   procedure Draw_Tacplot_Track_No
                               (Track_Location_X   : in Coordinate;
                                Track_Location_Y   : in Coordinate;
                                Track_No_String    : in String);
-- Tacplot Track Symbol Drawing Procedures
   procedure Draw_Tacplot_Unknown
                                (Track_Location_X  : in Coordinate;
                                 Track_Location_Y  : in Coordinate);
   procedure Draw_Tacplot_Friendly_Air
                               (Track_Location_X   : in Coordinate;
                                Track_Location_Y   : in Coordinate);
   procedure Draw_Tacplot_Hostile_Air
                               (Track_Location_X   : in Coordinate;
                                Track_Location_Y   : in Coordinate);
   procedure Draw_Tacplot_Unknown_Air
                               (Track_Location_X   : in Coordinate;
                                Track_Location_Y   : in Coordinate);
   procedure Draw_Tacplot_Neutral_Air
                               (Track_Location_X   : in Coordinate;
                                Track_Location_Y   : in Coordinate);
```

```
procedure Draw_Tacplot_Hostile_Surface
                                (Track_Location_X  : in Coordinate;
                                 Track_Location_Y  : in Coordinate);
procedure Draw_Tacplot_Friendly_Surface
                                (Track_Location_X  : in Coordinate;
                                 Track_Location_Y  : in Coordinate);
procedure Draw_Tacplot_Unknown_Surface
                                (Track_Location_X  : in Coordinate;
                                 Track_Location_Y  : in Coordinate);
procedure Draw_Tacplot_Neutral_Surface
                                (Track_Location_X  : in Coordinate;
                                 Track_Location_Y  : in Coordinate);
procedure Draw_Tacplot_Hostile_SubSurface
                                (Track_Location_X  : in Coordinate;
                                 Track_Location_Y  : in Coordinate);
procedure Draw_Tacplot_Neutral_SubSurface
                                (Track_Location_X  : in Coordinate;
                                 Track_Location_Y  : in Coordinate);
procedure Draw_Tacplot_Friendly_SubSurface
                                (Track_Location_X  : in Coordinate;
                                 Track_Location_Y  : in Coordinate);
procedure Draw_Tacplot_Unknown_SubSurface
                                (Track_Location_X  : in Coordinate;
                                 Track_Location_Y  : in Coordinate);
procedure Draw_Tacplot_DLRP     (Track_Location_X  : in Coordinate;
                                 Track_Location_Y  : in Coordinate);

procedure Draw_Tacplot_Ref_Point
                                (Track_Location_X  : in Coordinate;
                                 Track_Location_Y  : in Coordinate);
```

```
   procedure Draw_Tacplot_WayPoint
                                    (Track_Location_X  : in Coordinate;
                                     Track_Location_Y  : in Coordinate);
   procedure Draw_Tacplot_Nav_Hazard
                                    (Track_Location_X  : in Coordinate;
                                     Track_Location_Y  : in Coordinate);
   procedure Draw_Tacplot_Man_in_Water
                                    (Track_Location_X  : in Coordinate;
                                     Track_Location_Y  : in Coordinate);
--Track Info Write Procedures

   procedure Write_Tgt_Cse                (Course     : in String);
   procedure Write_Tgt_Spd                (Speed      : in String);
   procedure Write_Tgt_Id                 (Id         : in String);
   procedure Write_Tgt_Update_Hours       (Hours      : in String);
   procedure Write_Tgt_Update_Mins        (Minutes    : in String);
   procedure Write_Tgt_Brg                (Bearing    : in String);
   procedure Write_Tgt_Range              (Tgt_Range  : in String);
   procedure Write_Tgt_No                 (Number     : in String);
   procedure Write_Tgt_Ctl_Type           (Ctl_Type   : in String);
   procedure Write_Tgt_CPA_Bearing        (Bearing    : in String);
   procedure Write_Tgt_CPA_Range          (Tgt_Range  : in String);
   procedure Write_Tgt_CPA_Hours          (Hours      : in String);
   procedure Write_Tgt_CPA_Mins           (Minutes    : in String);
   procedure Write_Tgt_CPA_PNO;
   procedure Write_Tgt_CPA_CBDR;
   procedure Write_Tgt_Lat_Position       (Lat_Deg    : in String;
                                           Lat_Min    : in String;
                                           Lat_Sec    : in String;
                                           Lat_Dir    : in String);
```

```
   procedure Write_Tgt_Long_Position      (Long_Deg  : in String;
                                           Long_Min  : in String;
                                           Long_Sec  : in String;
                                           Long_Dir  : in String);
--Tacplot Write Procedures
   procedure Write_Hours                  (Hours     : in String);
   procedure Write_Mins                   (Minutes   : in String);
   procedure Write_Range_Scale            (Scale     : in String);
   procedure Write_Filters_Status         (Stat      : in String);
   procedure Write_Alerts_Status          (Stat      : in String);
--System Status Write Procedures
   procedure Write_Link_Status_Down;
   procedure Write_Gps_Status_Down;
   procedure Write_Radar_Status_Down;
   procedure Write_Sword_Status_Down;
   procedure Write_Gyro_Status_Down;
   procedure Write_Fath_Status_Down;
   procedure Write_Link_Status_Up;
   procedure Write_Gps_Status_Up;
   procedure Write_Radar_Status_Up;
   procedure Write_Sword_Status_Up;
   procedure Write_Gyro_Status_Up;
   procedure Write_Fath_Status_Up;
   procedure Write_Nav_Status             (Stat      : in String);
-- Intel Write Procedures
   procedure Write_Rel_Spd                (Rel_Spd   : in String);
   procedure Write_Rel_Brg                (Rel_Brg   : in String);
   procedure Write_Tgt_Angle              (Tgt_Angle: in String);
   procedure Write_Tgt_Name               (Tgt_Name  : in String);
   procedure Write_Tgt_Amp_Info           (Info      : in String);
   procedure Write_Tgt_Class              (Class     : in String);
```

```
   procedure Write_Tgt_Height                (Alt       : in String);
-- Alerts Write Procedures
   procedure Write_CPA_Alert                 (Target_ID : in String;
                                              CPA_Brg   : in String;
                                              CPA_Rng   : in String;
                                              CPA_Hrs   : in String;
                                              CPA_Mins  : in String);
-- Recommendations Write Procedures
   procedure Write_Intercept_Recommendation (Target_No : in String;
                                              Rec_Cse   : in String;
                                              Rec_Spd   : in String;
                                              Rec_Hrs   : in String;
                                              Rec_Mins  : in String);
end Draw_Display_Graphics;
package body Draw_Display_Graphics is
   procedure Clear_Interface_Displays is
      begin
         X_Clear_Window (Tacplot_Display, Tacplot_Window);
         X_Clear_Window (Track_Info_Display, Track_Info_Window);
         X_Clear_Window (Alerts_Display, Alerts_Window);
         X_Clear_Window (Intel_Display, Intel_Window);
      end Clear_Interface_Displays;

   procedure Draw_Tacplot_Hook
                          (Track_Location_X          : in Coordinate;
                           Track_Location_Y          : in Coordinate) is
     begin
       X_Draw_Arc (Tacplot_Display, Tacplot_Drawing_Space,
                   Tacplot_Display_Context,
                  (Track_Location_X - 20),
                  (Track_Location_Y - 20),
```

```
                Hook_Width, Hook_Height,
                0, 23039);
  end Draw_Tacplot_Hook;

procedure Draw_Tacplot_Speed_Leader
                        (Track_Location_X       : in Coordinate;
                         Track_Location_Y       : in Coordinate;
                         Speed_Leader_X         : in Coordinate;
                         Speed_Leader_Y         : in Coordinate) is
  begin
    X_Draw_Line
              (Tacplot_Display, Tacplot_Drawing_Space,
               Tacplot_Display_Context,
               Track_Location_X, Track_Location_Y,
              (Track_Location_X + Speed_Leader_X),
              (Track_Location_Y - Speed_Leader_Y));
  end Draw_Tacplot_Speed_Leader;
procedure Draw_Tacplot_Track_No
                       (Track_Location_X          : in Coordinate;
                        Track_Location_Y          : in Coordinate;
                        Track_No_String           : in String) is
      begin
        X_Draw_Image_String
              (Tacplot_Display, Tacplot_Drawing_Space,
               Tacplot_Display_Context,
              (Track_Location_X - 8),
              (Track_Location_Y + 20),
               Track_No_String);
    end Draw_Tacplot_Track_No;
```

```
        procedure Draw_Display_Attributes is
           begin
              X_Set_Foreground
                  (Tacplot_Display, Tacplot_Display_Context, 1);
              X_Draw_Arc
                   (Tacplot_Display, Tacplot_Drawing_Space,
                    Tacplot_Display_Context,
                    Tacplot_Window_X, Tacplot_Window_Y,
                   (Box_Width + 678), (Box_Height + 678), 0, 23039);
             X_Fill_Arc
                   (Tacplot_Display, Tacplot_Drawing_Space,
                    Tacplot_Display_Context,
                    Tacplot_Window_X, Tacplot_Window_Y,
                   (Box_Width + 678), (Box_Height + 678), 0, 23039);
---- Tacplot Text Defaults
              X_Draw_Image_String
                   (Tacplot_Display, Tacplot_Drawing_Space,
                    Tacplot_Display_Context,
                    5, 15,
                    "Scale:          ");
              X_Draw_Image_String
                   (Tacplot_Display, Tacplot_Drawing_Space,
                    Tacplot_Display_Context,
                    575, 15,
                   "Alerts :         ");
              X_Draw_Image_String
                   (Tacplot_Display, Tacplot_Drawing_Space,
                    Tacplot_Display_Context,
                    575, 30,
                   "Filters:         ");
```

```
            X_Draw_Image_String
                (Tacplot_Display, Tacplot_Drawing_Space,
                 Tacplot_Display_Context,
                 575, 45,
                "Intel  : Name");
            X_Draw_Image_String
                (Tacplot_Display, Tacplot_Drawing_Space,
                 Tacplot_Display_Context,
                 5, 690,
                "Set / Drift: 250 @ 1.5");
            X_Draw_Image_String
                (Tacplot_Display, Tacplot_Drawing_Space,
                 Tacplot_Display_Context,
                 5, 675,
                "True Wind  : 186 @ 6");
             X_Draw_Image_String
                (Tacplot_Display, Tacplot_Drawing_Space,
                 Tacplot_Display_Context,
                 5, 660,
                "Depth      : 2721 Ft");
             X_Draw_Image_String
                (Tacplot_Display, Tacplot_Drawing_Space,
                 Tacplot_Display_Context,
                 575, 690,
                "Time:              ");
X_Set_Foreground (Tacplot_Display, Tacplot_Display_Context, 0);
            X_Draw_Arc
                (Tacplot_Display, Tacplot_Drawing_Space,
                 Tacplot_Display_Context,
                (Centerpoint_X - 10), (Centerpoint_Y - 10),
```

```
                    Box_Width, Box_Height, 0, 23039);
                X_Draw_Line
                    (Tacplot_Display, Tacplot_Drawing_Space,
                     Tacplot_Display_Context,
                    (Centerpoint_X - 10), (Centerpoint_Y -  0),
                    (Centerpoint_X + 10), (Centerpoint_Y -  0));
                X_Draw_Line
                    (Tacplot_Display, Tacplot_Drawing_Space,
                     Tacplot_Display_Context,
                    (Centerpoint_X +  0), (Centerpoint_Y - 10),
                    (Centerpoint_X +  0), (Centerpoint_Y + 10));
--Draw Scale Selector Up/Down
                X_Draw_Rectangle
                    (Tacplot_Display, Tacplot_Drawing_Space,
                     Tacplot_Display_Context,
                     5, 25, 40, 30);
                 X_Fill_Rectangle
                    (Tacplot_Display, Tacplot_Drawing_Space,
                     Tacplot_Display_Context,
                     5, 25, 40, 30);
                X_Draw_Rectangle
                    (Tacplot_Display, Tacplot_Drawing_Space,
                     Tacplot_Display_Context,
                     45, 25, 40, 30);
            X_Set_Foreground (Tacplot_Display, Tacplot_Display_Context, 4);
                X_Fill_Rectangle
                    (Tacplot_Display, Tacplot_Drawing_Space,
                     Tacplot_Display_Context,
                     45, 25, 40, 30);
            X_Set_Foreground (Tacplot_Display, Tacplot_Display_Context, 0);
```

```
            X_Draw_Image_String
                (Tacplot_Display, Tacplot_Drawing_Space,
                 Tacplot_Display_Context,
                 57, 45,"Up");
        X_Set_Foreground (Tacplot_Display, Tacplot_Display_Context, 1);
            X_Draw_Image_String
                (Tacplot_Display, Tacplot_Drawing_Space,
                 Tacplot_Display_Context,
                 9, 45, "Down");
-- System Status Defaults
            X_Draw_Image_String
                (System_Status_Display, System_Status_Drawing_Space,
                 System_Status_Display_Context,
                 5, 242,
                "Navigation: ");
      end Draw_Display_Attributes;
--- Tacplot Drawing Procedures
      procedure Draw_Tacplot_Unknown
                    (Track_Location_X          : in Coordinate;
                     Track_Location_Y          : in Coordinate) is
        begin
          X_Draw_Arc
              (Tacplot_Display, Tacplot_Drawing_Space,
               Tacplot_Display_Context,
              (Track_Location_X - 10), (Track_Location_Y - 5),
               Box_Width, (Box_Height - 10), 0, 23039);
          X_Draw_Arc
              (Tacplot_Display, Tacplot_Drawing_Space,
               Tacplot_Display_Context,
              (Track_Location_X - 5), (Track_Location_Y - 10),
              (Box_Width - 10), Box_Height, 0, 23039);
```

```
  end Draw_Tacplot_Unknown;
procedure Draw_Tacplot_Friendly_Air
            (Track_Location_X          : in Coordinate;
             Track_Location_Y          : in Coordinate) is
   begin
       X_Draw_Arc
           (Tacplot_Display, Tacplot_Drawing_Space,
            Tacplot_Display_Context,
           (Track_Location_X - 10), (Track_Location_Y - 10),
            Box_Width, Box_Height,
            0, 11520);
end Draw_Tacplot_Friendly_Air;
procedure Draw_Tacplot_Hostile_Air
                (Track_Location_X          : in Coordinate;
                 Track_Location_Y          : in Coordinate) is
   begin
       X_Draw_Line
           (Tacplot_Display, Tacplot_Drawing_Space,
            Tacplot_Display_Context,
           (Track_Location_X - 10), (Track_Location_Y + 10),
            Track_Location_X, (Track_Location_Y - 10));
       X_Draw_Line
           (Tacplot_Display, Tacplot_Drawing_Space,
            Tacplot_Display_Context,
            Track_Location_X, (Track_Location_Y - 10),
           (Track_Location_X + 10), (Track_Location_Y + 10));
end Draw_Tacplot_Hostile_Air;
procedure Draw_Tacplot_Unknown_Air
            (Track_Location_X          : in Coordinate;
             Track_Location_Y          : in Coordinate) is
```

```
    begin
        X_Draw_Line
            (Tacplot_Display, Tacplot_Drawing_Space,
             Tacplot_Display_Context,
            (Track_Location_X - 10), (Track_Location_Y + 10),
            (Track_Location_X - 10), (Track_Location_Y - 10));
        X_Draw_Line
            (Tacplot_Display, Tacplot_Drawing_Space,
             Tacplot_Display_Context,
            (Track_Location_X - 10), (Track_Location_Y - 10),
            (Track_Location_X + 10), (Track_Location_Y - 10));
        X_Draw_Line
            (Tacplot_Display, Tacplot_Drawing_Space,
             Tacplot_Display_Context,
            (Track_Location_X + 10), (Track_Location_Y - 10),
            (Track_Location_X + 10), (Track_Location_Y + 10));
 end Draw_Tacplot_Unknown_Air;
procedure Draw_Tacplot_Neutral_Air
            (Track_Location_X          : in Coordinate;
             Track_Location_Y          : in Coordinate) is
   begin
        X_Draw_Line
            (Tacplot_Display, Tacplot_Drawing_Space,
             Tacplot_Display_Context,
            (Track_Location_X - 10), (Track_Location_Y + 10),
            (Track_Location_X - 10), (Track_Location_Y - 10));
        X_Draw_Line
            (Tacplot_Display, Tacplot_Drawing_Space,
             Tacplot_Display_Context,
            (Track_Location_X - 10), (Track_Location_Y - 10),
            (Track_Location_X + 10), (Track_Location_Y - 10));
```

```
            X_Draw_Line
                (Tacplot_Display, Tacplot_Drawing_Space,
                 Tacplot_Display_Context,
                (Track_Location_X + 10), (Track_Location_Y - 10),
                (Track_Location_X + 10), (Track_Location_Y + 10));
            X_Draw_Image_String
                (Tacplot_Display, Tacplot_Drawing_Space,
                 Tacplot_Display_Context,
                 Track_Location_X, Track_Location_Y,
                 Neutral_Symbol);
end Draw_Tacplot_Neutral_Air;
procedure Draw_Tacplot_Hostile_Surface
                (Track_Location_X          : in Coordinate;
                 Track_Location_Y          : in Coordinate) is
  begin
      X_Draw_Line
          (Tacplot_Display, Tacplot_Drawing_Space,
           Tacplot_Display_Context,
          (Track_Location_X - 10), (Track_Location_Y + 0),
          (Track_Location_X - 0), (Track_Location_Y - 10));
      X_Draw_Line
          (Tacplot_Display, Tacplot_Drawing_Space,
           Tacplot_Display_Context,
          (Track_Location_X - 0), (Track_Location_Y - 10),
          (Track_Location_X + 10), (Track_Location_Y - 0));
      X_Draw_Line
          (Tacplot_Display, Tacplot_Drawing_Space,
           Tacplot_Display_Context,
          (Track_Location_X + 10), (Track_Location_Y - 0),
          (Track_Location_X + 0), (Track_Location_Y + 10));
```

```
        X_Draw_Line
             (Tacplot_Display, Tacplot_Drawing_Space,
              Tacplot_Display_Context,
             (Track_Location_X + 0), (Track_Location_Y + 10),
             (Track_Location_X - 10), (Track_Location_Y + 0));
end Draw_Tacplot_Hostile_Surface;
procedure Draw_Tacplot_Friendly_Surface
              (Track_Location_X          : in Coordinate;
               Track_Location_Y          : in Coordinate) is
   begin
        X_Draw_Arc
             (Tacplot_Display, Tacplot_Drawing_Space,
              Tacplot_Display_Context,
             (Track_Location_X - 10), (Track_Location_Y - 10),
              Box_Width, Box_Height,
              0, 23039);
end Draw_Tacplot_Friendly_Surface;


procedure Draw_Tacplot_Unknown_Surface
              (Track_Location_X          : in Coordinate;
               Track_Location_Y          : in Coordinate) is
   begin
     X_Draw_Rectangle
             (Tacplot_Display, Tacplot_Drawing_Space,
              Tacplot_Display_Context,
             (Track_Location_X - 10), (Track_Location_Y - 10),
              Box_Width, Box_Height);
end Draw_Tacplot_Unknown_Surface;
```

```
procedure Draw_Tacplot_Neutral_Surface
             (Track_Location_X          : in Coordinate;
              Track_Location_Y          : in Coordinate) is
   begin
       X_Draw_Rectangle
            (Tacplot_Display, Tacplot_Drawing_Space,
             Tacplot_Display_Context,
            (Track_Location_X - 10), (Track_Location_Y - 10),
             Box_Width, Box_Height);
       X_Draw_Image_String
            (Tacplot_Display, Tacplot_Drawing_Space,
             Tacplot_Display_Context,
             Track_Location_X, Track_Location_Y,
             Neutral_Symbol);
end Draw_Tacplot_Neutral_Surface;
procedure Draw_Tacplot_Hostile_SubSurface
             (Track_Location_X          : in Coordinate;
              Track_Location_Y          : in Coordinate) is
   begin
       X_Draw_Line
            (Tacplot_Display, Tacplot_Drawing_Space,
             Tacplot_Display_Context,
            (Track_Location_X - 0), (Track_Location_Y + 10),
            (Track_Location_X - 10), (Track_Location_Y - 10));
       X_Draw_Line
            (Tacplot_Display, Tacplot_Drawing_Space,
             Tacplot_Display_Context,
            (Track_Location_X +  0), (Track_Location_Y + 10),
            (Track_Location_X + 10), (Track_Location_Y - 10));
       end Draw_Tacplot_Hostile_SubSurface;
```

```
procedure Draw_Tacplot_Neutral_SubSurface
                (Track_Location_X          : in Coordinate;
                 Track_Location_Y          : in Coordinate) is
   begin
       X_Draw_Line
            (Tacplot_Display, Tacplot_Drawing_Space,
             Tacplot_Display_Context,
            (Track_Location_X - 10), (Track_Location_Y - 10),
            (Track_Location_X - 10), (Track_Location_Y + 10));
       X_Draw_Line
            (Tacplot_Display, Tacplot_Drawing_Space,
             Tacplot_Display_Context,
            (Track_Location_X - 10), (Track_Location_Y + 10),
            (Track_Location_X + 10), (Track_Location_Y + 10));
       X_Draw_Line
            (Tacplot_Display, Tacplot_Drawing_Space,
             Tacplot_Display_Context,
            (Track_Location_X + 10), (Track_Location_Y + 10),
            (Track_Location_X + 10), (Track_Location_Y - 10));
        X_Draw_Image_String
           (Tacplot_Display, Tacplot_Drawing_Space,
            Tacplot_Display_Context,
            Track_Location_X, Track_Location_Y,
            Neutral_Symbol);
      end Draw_Tacplot_Neutral_SubSurface;
procedure Draw_Tacplot_Friendly_SubSurface
            (Track_Location_X          : in Coordinate;
             Track_Location_Y          : in Coordinate) is
   begin
     X_Draw_Arc
          (Tacplot_Display, Tacplot_Drawing_Space,
```

```
                Tacplot_Display_Context,
                (Track_Location_X - 10), (Track_Location_Y - 10),
                Box_Width, Box_Height,
                11520, 11520);
end Draw_Tacplot_Friendly_SubSurface;
procedure Draw_Tacplot_Unknown_SubSurface
                (Track_Location_X          : in Coordinate;
                 Track_Location_Y          : in Coordinate) is
   begin
     X_Draw_Line
           (Tacplot_Display, Tacplot_Drawing_Space,
            Tacplot_Display_Context,
           (Track_Location_X - 10), (Track_Location_Y - 10),
           (Track_Location_X - 10), (Track_Location_Y + 10));
     X_Draw_Line
           (Tacplot_Display, Tacplot_Drawing_Space,
            Tacplot_Display_Context,
           (Track_Location_X - 10), (Track_Location_Y + 10),
           (Track_Location_X + 10), (Track_Location_Y + 10));
     X_Draw_Line
           (Tacplot_Display, Tacplot_Drawing_Space,
            Tacplot_Display_Context,
           (Track_Location_X + 10), (Track_Location_Y + 10),
           (Track_Location_X + 10), (Track_Location_Y - 10));
   end Draw_Tacplot_Unknown_SubSurface;
procedure Draw_Tacplot_DLRP
                (Track_Location_X          : in Coordinate;
                 Track_Location_Y          : in Coordinate) is
   begin
     X_Draw_Rectangle
           (Tacplot_Display, Tacplot_Drawing_Space,
```

```
            Tacplot_Display_Context,
            (Track_Location_X - 10), (Track_Location_Y - 10),
            Box_Width, Box_Height);
      X_Draw_Line
            (Tacplot_Display, Tacplot_Drawing_Space,
            Tacplot_Display_Context,
            (Track_Location_X - 10), (Track_Location_Y - 10),
            (Track_Location_X + 10), (Track_Location_Y + 10));
     X_Draw_Line
            (Tacplot_Display, Tacplot_Drawing_Space,
            Tacplot_Display_Context,
            (Track_Location_X - 10), (Track_Location_Y + 10),
            (Track_Location_X + 10), (Track_Location_Y - 10));
end Draw_Tacplot_DLRP;
procedure Draw_Tacplot_Ref_Point
            (Track_Location_X          : in Coordinate;
             Track_Location_Y          : in Coordinate) is
  begin
    X_Draw_Line
            (Tacplot_Display, Tacplot_Drawing_Space,
            Tacplot_Display_Context,
            (Track_Location_X - 10), (Track_Location_Y - 10),
            (Track_Location_X + 10), (Track_Location_Y + 10));
    X_Draw_Line
            (Tacplot_Display, Tacplot_Drawing_Space,
            Tacplot_Display_Context,
            (Track_Location_X - 10), (Track_Location_Y + 10),
            (Track_Location_X + 10), (Track_Location_Y - 10));
end Draw_Tacplot_Ref_Point;
```

```
procedure Draw_Tacplot_WayPoint
            (Track_Location_X          : in Coordinate;
             Track_Location_Y          : in Coordinate) is
   begin
     X_Draw_Line
           (Tacplot_Display, Tacplot_Drawing_Space,
            Tacplot_Display_Context,
           (Track_Location_X - 10), (Track_Location_Y - 10),
           (Track_Location_X + 10), (Track_Location_Y + 10));
     X_Draw_Line
           (Tacplot_Display, Tacplot_Drawing_Space,
            Tacplot_Display_Context,
           (Track_Location_X - 10), (Track_Location_Y + 10),
           (Track_Location_X + 10), (Track_Location_Y - 10));
     X_Draw_Line
           (Tacplot_Display, Tacplot_Drawing_Space,
            Tacplot_Display_Context,
           (Track_Location_X + 10), (Track_Location_Y - 10),
           (Track_Location_X +  0), (Track_Location_Y - 10));
end Draw_Tacplot_WayPoint;
procedure Draw_Tacplot_Nav_Hazard
             (Track_Location_X          : in Coordinate;
              Track_Location_Y          : in Coordinate) is
   begin
     X_Draw_Line
           (Tacplot_Display, Tacplot_Drawing_Space,
            Tacplot_Display_Context,
           (Track_Location_X - 10), (Track_Location_Y -  0),
           (Track_Location_X + 10), (Track_Location_Y +  0));
     X_Draw_Line
           (Tacplot_Display, Tacplot_Drawing_Space,
```

```
                Tacplot_Display_Context,
                (Track_Location_X - 10), (Track_Location_Y -  0),
                (Track_Location_X -  5), (Track_Location_Y -  5));
          X_Draw_Line
                (Tacplot_Display, Tacplot_Drawing_Space,
                 Tacplot_Display_Context,
                (Track_Location_X + 10), (Track_Location_Y - 0),
                (Track_Location_X +  5), (Track_Location_Y + 5));
       end Draw_Tacplot_Nav_Hazard;
       procedure Draw_Tacplot_Man_in_Water
                  (Track_Location_X          : in Coordinate;
                   Track_Location_Y          : in Coordinate) is
          begin
            X_Draw_Arc
                  (Tacplot_Display, Tacplot_Drawing_Space,
                   Tacplot_Display_Context,
                  (Track_Location_X - 10), (Track_Location_Y - 10),
                  Box_Width, Box_Height, 0, 23039);
            X_Draw_Arc
                  (Tacplot_Display, Tacplot_Drawing_Space,
                   Tacplot_Display_Context,
                  (Track_Location_X - 5), (Track_Location_Y - 5),
                  (Box_Width / 2), (Box_Height / 2), 0, 23039);
       end Draw_Tacplot_Man_in_Water;
--- Track Information Write Procedures
        procedure Write_Tgt_Cse (Course:       in String) is
          begin
            X_Draw_Image_String
                  (Track_Info_Display, Track_Info_Drawing_Space,
                   Track_Info_Display_Context,
                   415, 15,
```

```
            "Target Course:      T");
        X_Draw_Image_String
              (Track_Info_Display, Track_Info_Drawing_Space,
               Track_Info_Display_Context,
               527, 15, Course);
    end Write_Tgt_Cse;
    procedure Write_Tgt_Spd (Speed:        in String) is
      begin
        X_Draw_Image_String
              (Track_Info_Display, Track_Info_Drawing_Space,
               Track_Info_Display_Context,
               415, 30,
              "Target Speed :      Kts");
        X_Draw_Image_String
              (Track_Info_Display, Track_Info_Drawing_Space,
               Track_Info_Display_Context,
               527, 30, Speed);
      end Write_Tgt_Spd;
  procedure Write_Tgt_Id (Id:        in String) is
      begin
        X_Draw_Image_String
              (Track_Info_Display, Track_Info_Drawing_Space,
               Track_Info_Display_Context,
               5, 80,
              "Target ID:");
        X_Draw_Image_String
              (Track_Info_Display, Track_Info_Drawing_Space,
               Track_Info_Display_Context,
               93, 80, Id);
      end Write_Tgt_Id;
```

```
procedure Write_Tgt_Update_Hours (Hours: in String) is
  begin
    X_Draw_Image_String
          (Track_Info_Display, Track_Info_Drawing_Space,
           Track_Info_Display_Context,
           5, 95,
          "Last Update:");
    X_Draw_Image_String
          (Track_Info_Display, Track_Info_Drawing_Space,
           Track_Info_Display_Context,
           109, 95, Hours);
  end Write_Tgt_Update_Hours;
procedure Write_Tgt_Update_Mins (Minutes:       in String) is
  begin
    X_Draw_Image_String
          (Track_Info_Display, Track_Info_Drawing_Space,
           Track_Info_Display_Context,
           133, 95, Minutes);
  end Write_Tgt_Update_Mins;
 procedure Write_Tgt_Brg (Bearing:       in String) is
 begin
    X_Draw_Image_String
          (Track_Info_Display, Track_Info_Drawing_Space,
           Track_Info_Display_Context,
           5, 15,
          "Target Bearing:         T");
    X_Draw_Image_String
          (Track_Info_Display, Track_Info_Drawing_Space,
           Track_Info_Display_Context,
           125, 15, Bearing);
  end Write_Tgt_Brg;
```

```
procedure Write_Tgt_Range (Tgt_Range:       in String) is
  begin
    X_Draw_Image_String
          (Track_Info_Display, Track_Info_Drawing_Space,
           Track_Info_Display_Context,
           5, 30,
          "Target Range:          Yards");
    X_Draw_Image_String
          (Track_Info_Display, Track_Info_Drawing_Space,
           Track_Info_Display_Context,
           125, 30, Tgt_Range);
  end Write_Tgt_Range;
procedure Write_Tgt_No (Number:       in String) is
  begin
    X_Draw_Image_String
          (Track_Info_Display, Track_Info_Drawing_Space,
           Track_Info_Display_Context,
           259, 15,
          "Contact:    ");
    X_Draw_Image_String
          (Track_Info_Display, Track_Info_Drawing_Space,
           Track_Info_Display_Context,
           339, 15, Number);
  end Write_Tgt_No;
procedure Write_Tgt_Ctl_Type (Ctl_Type:       in String) is
  begin
    X_Draw_Image_String
          (Track_Info_Display, Track_Info_Drawing_Space,
           Track_Info_Display_Context,
           328, 95,
          "Control Type:");
```

```
      X_Draw_Image_String
            (Track_Info_Display, Track_Info_Drawing_Space,
             Track_Info_Display_Context,
             440, 95, Ctl_Type);
    end Write_Tgt_Ctl_Type;
  procedure Write_Tgt_CPA_Bearing (Bearing:       in String) is
    begin
      X_Draw_Image_String
            (Track_Info_Display, Track_Info_Drawing_Space,
             Track_Info_Display_Context,
             183, 50,
            "Target CPA:     /        at      ");
      X_Draw_Image_String
            (Track_Info_Display, Track_Info_Drawing_Space,
             Track_Info_Display_Context,
             271, 50, Bearing);
    end Write_Tgt_CPA_Bearing;
  procedure Write_Tgt_CPA_Range (Tgt_Range:       in String) is
    begin
      X_Draw_Image_String
            (Track_Info_Display, Track_Info_Drawing_Space,
             Track_Info_Display_Context,
             318, 50, Tgt_Range);
    end Write_Tgt_CPA_Range;
  procedure Write_Tgt_CPA_Hours (Hours:       in String) is
    begin
      X_Draw_Image_String
            (Track_Info_Display, Track_Info_Drawing_Space,
             Track_Info_Display_Context,
             398, 50, Hours);
    end Write_Tgt_CPA_Hours;
```

```
 procedure Write_Tgt_CPA_Mins (Minutes:       in String) is
   begin
     X_Draw_Image_String
           (Track_Info_Display, Track_Info_Drawing_Space,
            Track_Info_Display_Context,
            423, 50, Minutes);
   end Write_Tgt_CPA_Mins;
procedure Write_Tgt_CPA_PNO is
   begin
     X_Set_Background (Track_Info_Display,
                       Track_Info_Display_Context, 14);
     X_Draw_Image_String
           (Track_Info_Display, Track_Info_Drawing_Space,
            Track_Info_Display_Context,
            231, 50, "Passed and Opening");
     X_Set_Background (Track_Info_Display,
                       Track_Info_Display_Context, 7);
   end Write_Tgt_CPA_PNO;
procedure Write_Tgt_CPA_CBDR is
   begin
     X_Set_Background (Track_Info_Display,
                       Track_Info_Display_Context, 12);
     X_Draw_Image_String
           (Track_Info_Display, Track_Info_Drawing_Space,
            Track_Info_Display_Context,
            231, 50, "     **** CBDR ****    ");
     X_Set_Background (Track_Info_Display,
                       Track_Info_Display_Context, 7);
   end Write_Tgt_CPA_CBDR;
```

```
procedure Write_Tgt_Lat_Position        (Lat_Deg: in String;
                                         Lat_Min: in String;
                                         Lat_Sec: in String;
                                         Lat_Dir: in String) is
   begin
      X_Draw_Image_String
           (Track_Info_Display, Track_Info_Drawing_Space,
           Track_Info_Display_Context,
           142, 65,
           "Target Position: ");
      X_Draw_Image_String
           (Track_Info_Display, Track_Info_Drawing_Space,
           Track_Info_Display_Context,
           278, 65, Lat_Deg);
      X_Draw_Image_String
           (Track_Info_Display, Track_Info_Drawing_Space,
           Track_Info_Display_Context,
           302, 65, Lat_Min);
      X_Draw_Image_String
           (Track_Info_Display, Track_Info_Drawing_Space,
           Track_Info_Display_Context,
           326, 65, Lat_Sec);
      X_Draw_Image_String
           (Track_Info_Display, Track_Info_Drawing_Space,
           Track_Info_Display_Context,
           356, 65, Lat_Dir);
end Write_Tgt_Lat_Position;
 procedure Write_Tgt_Long_Position       (Long_Deg: in String;
                                          Long_Min: in String;
                                          Long_Sec: in String;
                                          Long_Dir: in String) is
```

```
        begin
           X_Draw_Image_String
                (Track_Info_Display, Track_Info_Drawing_Space,
                Track_Info_Display_Context,
                372, 65, "/");
           X_Draw_Image_String
                (Track_Info_Display, Track_Info_Drawing_Space,
                Track_Info_Display_Context,
                388, 65, Long_Deg);
           X_Draw_Image_String
                (Track_Info_Display, Track_Info_Drawing_Space,
                Track_Info_Display_Context,
                420, 65, Long_Min);
           X_Draw_Image_String
                (Track_Info_Display, Track_Info_Drawing_Space,
                Track_Info_Display_Context,
                444, 65, Long_Sec);
           X_Draw_Image_String
                (Track_Info_Display, Track_Info_Drawing_Space,
                Track_Info_Display_Context,
                476, 65, Long_Dir);
      end Write_Tgt_Long_Position;
--- Tacplot Write Procedures
         procedure Write_Hours (Hours:       in String) is
         begin
           X_Set_Foreground
             (Tacplot_Display, Tacplot_Display_Context, 1);
              X_Draw_Image_String
              (Tacplot_Display, Tacplot_Drawing_Space,
              Tacplot_Display_Context,
              617, 690, Hours);
```

```
      X_Set_Foreground
        (Tacplot_Display, Tacplot_Display_Context, 0);
    end Write_Hours;
    procedure Write_Mins (Minutes:       in String) is
    begin
      X_Set_Foreground
       (Tacplot_Display, Tacplot_Display_Context, 1);
      X_Draw_Image_String
           (Tacplot_Display, Tacplot_Drawing_Space,
           Tacplot_Display_Context,
           641, 690, Minutes);
      X_Set_Foreground
       (Tacplot_Display, Tacplot_Display_Context, 0);
     end Write_Mins;
procedure Write_Range_Scale (Scale:         in String) is
  begin
    X_Draw_Image_String
           (Tacplot_Display, Tacplot_Drawing_Space,
           Tacplot_Display_Context,
           53, 15, Scale);
    X_Draw_Image_String
           (Tacplot_Display, Tacplot_Drawing_Space,
           Tacplot_Display_Context,
           101, 15, "Miles");
end Write_Range_Scale;
procedure Write_Filters_Status (Stat:        in String) is
  begin
    X_Draw_Image_String
           (Tacplot_Display, Tacplot_Drawing_Space,
           Tacplot_Display_Context,
           647, 30, Stat);
```

```
        end Write_Filters_Status;
         procedure Write_Alerts_Status (Stat:        in String) is
          begin
            X_Draw_Image_String
                   (Tacplot_Display, Tacplot_Drawing_Space,
                   Tacplot_Display_Context,
                   647, 15, Stat);
        end Write_Alerts_Status;
-- System Status Write Procedures
         procedure Write_Gps_Status_Down is
           begin
            X_Set_Background
              (System_Status_Display, System_Status_Display_Context, 1);
            X_Set_Foreground
            (System_Status_Display, System_Status_Display_Context, 12);
            X_Draw_Image_String
                   (System_Status_Display, System_Status_Drawing_Space,
                   System_Status_Display_Context,
                   101, 15,
                   "DOWN");
            X_Set_Background
             (System_Status_Display, System_Status_Display_Context, 14);
            X_Set_Foreground
            (System_Status_Display, System_Status_Display_Context, 1);
         end Write_Gps_Status_Down;
        procedure Write_Radar_Status_Down is
           begin
            X_Set_Background
              (System_Status_Display, System_Status_Display_Context, 1);
            X_Set_Foreground
           (System_Status_Display, System_Status_Display_Context, 12);
```

```
   X_Draw_Image_String
         (System_Status_Display, System_Status_Drawing_Space,
         System_Status_Display_Context,
         101, 30,
         "DOWN");
   X_Set_Background
    (System_Status_Display, System_Status_Display_Context, 14);
   X_Set_Foreground
   (System_Status_Display, System_Status_Display_Context, 1);
 end Write_Radar_Status_Down;
procedure Write_Link_Status_Down is
begin
  X_Set_Background
    (System_Status_Display, System_Status_Display_Context, 1);
  X_Set_Foreground
 (System_Status_Display, System_Status_Display_Context, 12);
  X_Draw_Image_String
        (System_Status_Display, System_Status_Drawing_Space,
        System_Status_Display_Context,
        101, 45,
        "DOWN");
  X_Set_Background
   (System_Status_Display, System_Status_Display_Context, 14);
  X_Set_Foreground
  (System_Status_Display, System_Status_Display_Context, 1);
 end Write_Link_Status_Down;
procedure Write_Fath_Status_Down is
begin
  X_Set_Background
    (System_Status_Display, System_Status_Display_Context, 1);
  X_Set_Foreground
```

```
   (System_Status_Display, System_Status_Display_Context, 12);
    X_Draw_Image_String
          (System_Status_Display, System_Status_Drawing_Space,
          System_Status_Display_Context,
          346, 15,
          "DOWN");
    X_Set_Background
     (System_Status_Display, System_Status_Display_Context, 14);
    X_Set_Foreground
    (System_Status_Display, System_Status_Display_Context, 1);
  end Write_Fath_Status_Down;
procedure Write_Gyro_Status_Down is
begin
   X_Set_Background
     (System_Status_Display, System_Status_Display_Context, 1);
   X_Set_Foreground
  (System_Status_Display, System_Status_Display_Context, 12);
   X_Draw_Image_String
          (System_Status_Display, System_Status_Drawing_Space,
          System_Status_Display_Context,
          346, 30,
          "DOWN");
   X_Set_Background
    (System_Status_Display, System_Status_Display_Context, 14);
   X_Set_Foreground
   (System_Status_Display, System_Status_Display_Context, 1);
 end Write_Gyro_Status_Down;
procedure Write_Sword_Status_Down is
begin
  X_Set_Background
    (System_Status_Display, System_Status_Display_Context, 1);
```

```
    X_Set_Foreground
   (System_Status_Display, System_Status_Display_Context, 12);
    X_Draw_Image_String
          (System_Status_Display, System_Status_Drawing_Space,
          System_Status_Display_Context,
          346, 45,
          "DOWN");
    X_Set_Background
      (System_Status_Display, System_Status_Display_Context, 14);
    X_Set_Foreground
    (System_Status_Display, System_Status_Display_Context, 1);
  end Write_Sword_Status_Down;
procedure Write_Gps_Status_Up is
  begin
    X_Set_Background
      (System_Status_Display, System_Status_Display_Context, 13);
    X_Set_Foreground
    (System_Status_Display, System_Status_Display_Context, 1);
    X_Draw_Image_String
          (System_Status_Display, System_Status_Drawing_Space,
          System_Status_Display_Context,
          101, 15,
          "UP");
    X_Set_Background
      (System_Status_Display, System_Status_Display_Context, 14);
    X_Set_Foreground
    (System_Status_Display, System_Status_Display_Context, 1);
  end Write_Gps_Status_Up;
  procedure Write_Radar_Status_Up is
  begin
    X_Set_Background
```

```
    (System_Status_Display, System_Status_Display_Context, 13);
   X_Set_Foreground
   (System_Status_Display, System_Status_Display_Context, 1);
   X_Draw_Image_String
         (System_Status_Display, System_Status_Drawing_Space,
         System_Status_Display_Context,
         101, 30,
         "UP");
   X_Set_Background
    (System_Status_Display, System_Status_Display_Context, 14);
   X_Set_Foreground
   (System_Status_Display, System_Status_Display_Context, 1);
 end Write_Radar_Status_Up;
procedure Write_Link_Status_Up is
begin
  X_Set_Background
   (System_Status_Display, System_Status_Display_Context, 13);
  X_Set_Foreground
  (System_Status_Display, System_Status_Display_Context, 1);
  X_Draw_Image_String
        (System_Status_Display, System_Status_Drawing_Space,
        System_Status_Display_Context,
        101, 45,
        "UP");
 X_Set_Background
  (System_Status_Display, System_Status_Display_Context, 14);
 X_Set_Foreground
 (System_Status_Display, System_Status_Display_Context, 1);
end Write_Link_Status_Up;
```

```
procedure Write_Fath_Status_Up is
begin
  X_Set_Background
    (System_Status_Display, System_Status_Display_Context, 13);
  X_Set_Foreground
  (System_Status_Display, System_Status_Display_Context, 1);
  X_Draw_Image_String
        (System_Status_Display, System_Status_Drawing_Space,
        System_Status_Display_Context,
        346, 15,
        "UP");
  X_Set_Background
    (System_Status_Display, System_Status_Display_Context, 14);
  X_Set_Foreground
  (System_Status_Display, System_Status_Display_Context, 1);
 end Write_Fath_Status_Up;
procedure Write_Gyro_Status_Up is
begin
  X_Set_Background
    (System_Status_Display, System_Status_Display_Context, 13);
  X_Set_Foreground
  (System_Status_Display, System_Status_Display_Context, 1);
  X_Draw_Image_String
        (System_Status_Display, System_Status_Drawing_Space,
        System_Status_Display_Context,
        346, 30,
        "UP");
  X_Set_Background
    (System_Status_Display, System_Status_Display_Context, 14);
  X_Set_Foreground
  (System_Status_Display, System_Status_Display_Context, 1);
```

```
    end Write_Gyro_Status_Up;
  procedure Write_Sword_Status_Up is
  begin
    X_Set_Background
     (System_Status_Display, System_Status_Display_Context, 13);
    X_Set_Foreground
    (System_Status_Display, System_Status_Display_Context, 1);
    X_Draw_Image_String
          (System_Status_Display, System_Status_Drawing_Space,
          System_Status_Display_Context,
          346, 45,
          "UP");
    X_Set_Background
    (System_Status_Display, System_Status_Display_Context, 14);
    X_Set_Foreground
    (System_Status_Display, System_Status_Display_Context, 1);
   end Write_Sword_Status_Up;
procedure Write_Nav_Status (Stat: in String) is
  begin
    X_Set_Background
     (System_Status_Display, System_Status_Display_Context, 13);
    X_Set_Foreground
    (System_Status_Display, System_Status_Display_Context, 1);
    X_Draw_Image_String
          (System_Status_Display, System_Status_Drawing_Space,
          System_Status_Display_Context,
          101, 242, Stat);
    X_Set_Background
     (System_Status_Display, System_Status_Display_Context, 14);
    X_Set_Foreground
    (System_Status_Display, System_Status_Display_Context, 1);
```

```
            end Write_Nav_Status;
--- Intel Write Procedures
      procedure Write_Rel_Brg (Rel_Brg:              in String) is
         begin
            X_Draw_Image_String
                  (Intel_Display, Intel_Drawing_Space,
                  Intel_Display_Context,
                  5, 15,
                  "Contact Relative Bearing:    ");
             X_Draw_Image_String
                  (Intel_Display, Intel_Drawing_Space,
                  Intel_Display_Context,
                  213, 15, Rel_Brg);
           X_Draw_Image_String
                  (Intel_Display, Intel_Drawing_Space,
                  Intel_Display_Context,
                  253, 15, "Degrees");
        end Write_Rel_Brg;
 -- Intel / Relative Contact Info
       procedure Write_Rel_Spd (Rel_Spd: in String) is
         begin
           X_Draw_Image_String
                  (Intel_Display, Intel_Drawing_Space,
                  Intel_Display_Context,
                  5, 30,
                  "Contact Relative Speed  :    ");
           X_Draw_Image_String
                  (Intel_Display, Intel_Drawing_Space,
                  Intel_Display_Context,
                  213, 30, Rel_Spd);
           X_Draw_Image_String
```

```
                (Intel_Display, Intel_Drawing_Space,
                Intel_Display_Context,
                253, 30, "Knots");
        end Write_Rel_Spd;
    procedure Write_Tgt_Angle (Tgt_Angle: in String) is
       begin
         X_Draw_Image_String
                (Intel_Display, Intel_Drawing_Space,
                Intel_Display_Context,
                5, 160,
                "Target Angle:    ");
         X_Draw_Image_String
                (Intel_Display, Intel_Drawing_Space,
                Intel_Display_Context,
                117, 160, Tgt_Angle);
         X_Draw_Image_Str---
                (Intel_Display, Intel_Drawing_Space,
                Intel_Display_Context,
                157, 160, "Degrees");
      end Write_Tgt_Angle;
procedure Write_Tgt_Name (Tgt_Name : in String) is
  begin
         X_Draw_Image_String
                (Intel_Display, Intel_Drawing_Space,
                Intel_Display_Context,
                5, 60, "Track Name: ");
         X_Draw_Image_String
                (Intel_Display, Intel_Drawing_Space,
                Intel_Display_Context,
                109, 60, Tgt_Name);
    end Write_Tgt_Name;
```

```
procedure Write_Tgt_Class (Class : in String) is
  begin
        X_Draw_Image_String
                (Intel_Display, Intel_Drawing_Space,
                Intel_Display_Context,
                5, 75, "Track Class: ");
        X_Draw_Image_String
                (Intel_Display, Intel_Drawing_Space,
                Intel_Display_Context,
                110, 75, Class);
   end Write_Tgt_Class;
procedure Write_Tgt_Amp_Info (Info : in String) is
  begin
        X_Draw_Image_String
                (Intel_Display, Intel_Drawing_Space,
                Intel_Display_Context,
                5, 90, "Track Info: ");
        X_Draw_Image_String
                (Intel_Display, Intel_Drawing_Space,
                Intel_Display_Context,
                109, 90, Info);
   end Write_Tgt_Amp_Info;
procedure Write_Tgt_Height (Alt : in String) is
  begin
        X_Draw_Image_String
                (Intel_Display, Intel_Drawing_Space,
                Intel_Display_Context,
                5, 105, "Track Altitude: ");
        X_Draw_Image_String
                (Intel_Display, Intel_Drawing_Space,
                Intel_Display_Context,
```

```
                136, 105, Alt);
            X_Draw_Image_String
                (Intel_Display, Intel_Drawing_Space,
                Intel_Display_Context,
                192, 105, "Feet");
        end Write_Tgt_Height;
-- Alerts Write Procedures
        procedure Write_CPA_Alert (Target_ID          : in String;
                                   CPA_Brg            : in String;
                                   CPA_Rng            : in String;
                                   CPA_Hrs            : in String;
                                   CPA_Mins           : in String) is
          begin
            X_Set_Foreground
               (Alerts_Display, Alerts_Display_Context, 12);
            X_Fill_Rectangle
                (Alerts_Display, Alerts_Drawing_Space,
                Alerts_Display_Context,
                0, 0, 433, 90);
            X_Set_Foreground
               (Alerts_Display, Alerts_Display_Context, 1);
            X_Draw_Image_String
                (Alerts_Display, Alerts_Drawing_Space,
                Alerts_Display_Context,
                1, 45, " Contact");
            X_Draw_Image_String
                (Alerts_Display, Alerts_Drawing_Space,
                Alerts_Display_Context,
                65, 45, Target_ID);
            X_Draw_Image_String
                (Alerts_Display, Alerts_Drawing_Space,
```

```
                Alerts_Display_Context,
                81, 45, " :  CPA");
        X_Draw_Image_String
                (Alerts_Display, Alerts_Drawing_Space,
                Alerts_Display_Context,
                139, 45, CPA_Brg);
        X_Draw_Image_String
                (Alerts_Display, Alerts_Drawing_Space,
                Alerts_Display_Context,
                179, 45, "/");
        X_Draw_Image_String
                (Alerts_Display, Alerts_Drawing_Space,
                Alerts_Display_Context,
                187, 45, CPA_Rng);
        X_Draw_Image_String
                (Alerts_Display, Alerts_Drawing_Space,
                Alerts_Display_Context,
                245, 45, "Yards at Time :");
        X_Draw_Image_String
                (Alerts_Display, Alerts_Drawing_Space,
                Alerts_Display_Context,
                355, 45, CPA_Hrs);
        X_Draw_Image_String
                (Alerts_Display, Alerts_Drawing_Space,
                Alerts_Display_Context,
                379, 45, CPA_Mins);
end Write_CPA_Alert;
```

```
procedure Write_Intercept_Recommendation
                                (Target_No          : in String;
                                 Rec_Cse            : in String;
                                 Rec_Spd            : in String;
                                 Rec_Hrs            : in String;
                                 Rec_Mins           : in String) is
   begin
     X_Draw_Image_String
        (Recommendations_Display, Recommendations_Drawing_Space,
         Recommendations_Display_Context,
         5, 15,
         "Intercept to Target #");
     X_Draw_Image_String
          (Recommendations_Display, Recommendations_Drawing_Space,
           Recommendations_Display_Context,
           181, 15,
           Target_No);
    X_Draw_Image_String
          (Recommendations_Display, Recommendations_Drawing_Space,
           Recommendations_Display_Context,
           5, 30,
           "-------------***-------------");
     X_Draw_Image_String
          (Recommendations_Display, Recommendations_Drawing_Space,
           Recommendations_Display_Context,
           5, 45,
           "Intercept Course:");
     X_Draw_Image_String
          (Recommendations_Display, Recommendations_Drawing_Space,
           Recommendations_Display_Context,
           149, 45,
```

```
                Rec_Cse);
            X_Draw_Image_String
                (Recommendations_Display, Recommendations_Drawing_Space,
                 Recommendations_Display_Context,
                 5, 60,
                 "Intercept Speed :");
            X_Draw_Image_String
                (Recommendations_Display, Recommendations_Drawing_Space,
                 Recommendations_Display_Context,
                 149, 60,
                 Rec_Spd);
            X_Draw_Image_String
                (Recommendations_Display, Recommendations_Drawing_Space,
                 Recommendations_Display_Context,
                 5, 75,
                 "Intercept Time  :");
            X_Draw_Image_String
                (Recommendations_Display, Recommendations_Drawing_Space,
                 Recommendations_Display_Context,
                 149, 75,
                 Rec_Hrs);
            X_Draw_Image_String
                (Recommendations_Display, Recommendations_Drawing_Space,
                 Recommendations_Display_Context,
                 173, 75,
                 Rec_Mins);
        end Write_Intercept_Recommendation;
    end Draw_Display_Graphics;
```

APPENDIX E

TACPLOT SET PACKAGE

```
--        Author: Michael G. Stockwell
--        Date  : 12 September 1991
--        Description: Opens and reads generic set package for Track
                       objects


with TRACK_PKG; use TRACK_PKG;
with PROCESS_Interface_Displays; use PROCESS_Interface_Displays;
with GENERIC_SET_PKG;

package TACPLOT_PKG is

  type TACPLOT  is private;

  procedure EMPTY_TACPLOT;
  procedure ADD_TACPLOT_ELEMENT          (Element: in Track);
  procedure DISPLAY_TACPLOT;

  package TACPLOT_ELEMENT_SET_PKG is
                  new GENERIC_SET_PKG (T => TRACK, Block_Size => 2);
private
  type TACPLOT is
     record
```

```
        S : TACPLOT_ELEMENT_SET_PKG.SET;
      end record;
end TACPLOT_PKG;
package body TACPLOT_PKG is

   tacplot_set : tacplot;
   element     : track;

   procedure DRAW_TACPLOT_ELEMENTS (Current_Track : in Track) is
     begin
       Process_Tacplot_Objects (Current_Track);
     end DRAW_TACPLOT_ELEMENTS;

   procedure SCAN_TACPLOT is new TACPLOT_ELEMENT_SET_PKG.GENERIC_SCAN
                              (generate => draw_tacplot_elements);

   procedure EMPTY_TACPLOT is
     begin
       if TACPLOT_ELEMENT_SET_PKG.SIZE(tacplot_set.s) /= 0 then
          TACPLOT_ELEMENT_SET_PKG.EMPTY (tacplot_set.s);
        end if;
end EMPTY_TACPLOT;

  procedure ADD_TACPLOT_ELEMENT  (element: in Track) is
    begin
      TACPLOT_ELEMENT_SET_PKG.ADD (element, tacplot_set.s);
    end ADD_TACPLOT_ELEMENT;

  procedure DISPLAY_TACPLOT is
    begin
      SCAN_TACPLOT (tacplot_set.s);
```

```
    end DISPLAY_TACPLOT;

end TACPLOT_PKG;
```

APPENDIX F

MAIN PROCEDURE

```
with X_lib; use X_Lib;
with tae; use tae;
with X_Windows;
with TEXT_IO, TRACK_PKG, GLOBAL_POSITION_PKG, GLOBAL_OBSERVATION_PKG,
     VELOCITY_PKG, ANGLE_PKG, SPEED_PKG, RELATIVE_TIME_PKG, TACPLOT_PKG,
     ABSOLUTE_TIME_PKG, DISTANCE_PKG, DIRECT_IO, TRACK_DATABASE_PKG,
     CPA_PKG, RELATIVE_POSITION_PKG, VECTOR_2_PKG, FILTER_PKG,
     SYSTEM_STATUS_PKG, NAVIGATION_PKG, INTEGRATION_SYSTEM_PKG,
     MONITOR_DISPLAYS, OPEN_INTERFACE_DISPLAYS,
     PROCESS_INTERFACE_DISPLAYS,
     DRAW_DISPLAY_GRAPHICS, SYSTEM_STATUS_PKG, INTERCEPT_PKG;

 use TEXT_IO, TRACK_PKG, GLOBAL_POSITION_PKG, GLOBAL_OBSERVATION_PKG,
     VELOCITY_PKG, ANGLE_PKG, SPEED_PKG, RELATIVE_TIME_PKG, TACPLOT_PKG,
     ABSOLUTE_TIME_PKG, DISTANCE_PKG, TRACK_DATABASE_PKG, CPA_PKG,
     RELATIVE_POSITION_PKG, VECTOR_2_PKG, FILTER_PKG,
     SYSTEM_STATUS_PKG, NAVIGATION_PKG, INTEGRATION_SYSTEM_PKG,
     MONITOR_DISPLAYS, OPEN_INTERFACE_DISPLAYS,
     PROCESS_INTERFACE_DISPLAYS,
     DRAW_DISPLAY_GRAPHICS, SYSTEM_STATUS_PKG, INTERCEPT_PKG;
```

```
procedure menu is
   OWN_OBS              : GLOBAL_OBSERVATION;
   OWN_CRS              : Angle_Pkg.ANGLE;
   OWN_SPD              : SPEED;
   OWN_CRS_SPD          : VELOCITY;
   ADD_TRACK_FLAG       : BOOLEAN              := FALSE;
   MOD_TRACK_FLAG       : BOOLEAN              := FALSE;
   MOD_TRACK            : NATURAL              := 9999;
   DELETE_TRACK         : NATURAL              := 9999;
   TARGET_BEARING       : ANGLE_PKG.ANGLE      := 0.0;
   TARGET_RANGE         : DISTANCE             := 5000.0;
   TARGET_OBS           : GLOBAL_OBSERVATION;
   TGT_BRG_RNG          : RELATIVE_POSITION;
   TRK                  : TRACK;
   OWNSHIP              : TRACK;
   LAT_DEG              : NATURAL              := 0;
   LAT_MIN              : NATURAL              := 0;
   LAT_SEC              : NATURAL              := 0;
   LAT_DIR              : NORTH_SOUTH          := N;
   LONG_DEG             : NATURAL              := 0;
   LONG_MIN             : NATURAL              := 0;
   LONG_SEC             : NATURAL              := 0;
   LONG_DIR             : EAST_WEST            := E;
   ALERT_RANGE          : FLOAT                := 3000.0;
   GPS_SYS              : BOOLEAN              := FALSE;
   RADAR_SYS            : BOOLEAN              := FALSE;
   LINK_SYS             : BOOLEAN              := FALSE;
   GYRO_SYS             : BOOLEAN              := FALSE;
   FATH_SYS             : BOOLEAN              := FALSE;
   PIT_SYS              : BOOLEAN              := FALSE;
   EQPT_STATUS          : STATUS;
```

```
   TRACK_ID            : IDENTITY_TYPE;
   TRACK_CAT           : TRACK_CATEGORY;
   TRACK_CSE           : ANGLE_PKG.ANGLE;
   TRACK_SPD           : SPEED;
   STATION_TIME        : NATURAL              := 15;
   STATION_SPEED       : FLOAT                := 25.0;
   INTERCEPT_TRACK_NO  : NATURAL              := 9999;
   INTERCEPT_TRACK     : TRACK;
   INTERCEPT_BY_TIME   : BOOLEAN              := TRUE;
   INTERCEPT_BY_SPEED  : BOOLEAN              := FALSE;
   T_CAT               : TRACK_CATEGORY       := AIR_PLATFORM;
   RID                 : RELATION_ID          := LESS_OR_EQUAL;
   RG                  : DISTANCE             := 100000.0;
   FC                  : FILTER_CATEGORY;
   DA                  : DISTANCE_ATTRIBUTE_ID;
   PID                 : IDENTITY_TYPE;
   EQ                  : EQUALITY_RELATION_ID := EQUAL;

package menu_support is
    package taefloat_io is new text_io.float_io (taefloat);
    procedure initializePanels (file : in string);
    -- BEGIN EVENT_HANDLERs
    procedure menu_track_button   (info : in tae_wpt.event_context_ptr);
    procedure menu_plots_menu     (info : in tae_wpt.event_context_ptr);
    procedure menu_alerts_button  (info : in tae_wpt.event_context_ptr);
    procedure menu_filters_button(info : in tae_wpt.event_context_ptr);
    procedure menu_defaults_button
                                  (info : in tae_wpt.event_context_ptr);
    procedure menu_intel_button   (info : in tae_wpt.event_context_ptr);
    procedure menu_nav_button     (info : in tae_wpt.event_context_ptr);
    procedure menu_lists_button   (info : in tae_wpt.event_context_ptr);
```

```
procedure menu_coding_button (info : in tae_wpt.event_context_ptr);
procedure Sorry_OK_Button    (info : in tae_wpt.event_context_ptr);
procedure Sorry_OK_2         (info : in tae_wpt.event_context_ptr);
procedure trk_sel_NoName01   (info : in tae_wpt.event_context_ptr);
procedure trk_sel_trk_sel_cancel
                             (info : in tae_wpt.event_context_ptr);
procedure trk_sel_trk_sel_lat_lon
                             (info : in tae_wpt.event_context_ptr);
procedure trk_sel_trk_sel_brg_rng
                             (info : in tae_wpt.event_context_ptr);
procedure trk_sel_trk_sel_screen
                             (info : in tae_wpt.event_context_ptr);
procedure sel_brg_tgt_bearing(info : in tae_wpt.event_context_ptr);
procedure sel_brg_sel_brg_canx_1
                             (info : in tae_wpt.event_context_ptr);
procedure sel_brg_Enter_Data (info : in tae_wpt.event_context_ptr);
procedure sel_brg_sel_brg_canx
                             (info : in tae_wpt.event_context_ptr);
procedure sel_brg_tgt_range  (info : in tae_wpt.event_context_ptr);
procedure glob_pos_lat_min   (info : in tae_wpt.event_context_ptr);
procedure glob_pos_lat_sec   (info : in tae_wpt.event_context_ptr);
procedure glob_pos_long_deg  (info : in tae_wpt.event_context_ptr);
procedure glob_pos_long_min  (info : in tae_wpt.event_context_ptr);
procedure glob_pos_long_sec  (info : in tae_wpt.event_context_ptr);
procedure glob_pos_lat_deg   (info : in tae_wpt.event_context_ptr);
procedure glob_pos_spacer    (info : in tae_wpt.event_context_ptr);
procedure glob_pos_Spacer_1  (info : in tae_wpt.event_context_ptr);
procedure glob_pos_lat_dir   (info : in tae_wpt.event_context_ptr);
procedure glob_pos_long_dir  (info : in tae_wpt.event_context_ptr);
procedure confirm_conf_continue
                             (info : in tae_wpt.event_context_ptr);
```

```
procedure confirm_conf_cancel
                                (info : in tae_wpt.event_context_ptr);
procedure alrt_sel_enter_data
                                (info : in tae_wpt.event_context_ptr);
procedure alrt_sel_cpa_alrt_range
                                (info : in tae_wpt.event_context_ptr);
procedure alrt_sel_alrt_canx_1
                                (info : in tae_wpt.event_context_ptr);
procedure alrt_sel_alrt_canx_2
                                (info : in tae_wpt.event_context_ptr);
procedure eqpt_sel_gps_sel    (info : in tae_wpt.event_context_ptr);
procedure eqpt_sel_radar_sel (info : in tae_wpt.event_context_ptr);
procedure eqpt_sel_link_sel   (info : in tae_wpt.event_context_ptr);
procedure eqpt_sel_gyro_sel   (info : in tae_wpt.event_context_ptr);
procedure eqpt_sel_fath_sel   (info : in tae_wpt.event_context_ptr);
procedure eqpt_sel_pit_sel    (info : in tae_wpt.event_context_ptr);
procedure eqpt_sel_enter_data(info : in tae_wpt.event_context_ptr);
procedure title_exit_system   (info : in tae_wpt.event_context_ptr);
procedure exit_ok_exit_continue
                                (info : in tae_wpt.event_context_ptr);
procedure exit_ok_exit_cancel(info : in tae_wpt.event_context_ptr);
procedure trk_mod_trk_identity
                                (info : in tae_wpt.event_context_ptr);
procedure trk_mod_trk_control(info : in tae_wpt.event_context_ptr);
procedure trk_mod_trk_point_type
                                (info : in tae_wpt.event_context_ptr);
procedure trk_mod_trk_category
                                (info : in tae_wpt.event_context_ptr);
procedure trk_mod_trk_position
                                (info : in tae_wpt.event_context_ptr);
procedure trk_mod_trk_amp_info
```

```
                                (info : in tae_wpt.event_context_ptr);
procedure trk_mod_trk_name    (info : in tae_wpt.event_context_ptr);
procedure trk_mod_trk_course (info : in tae_wpt.event_context_ptr);
procedure trk_mod_trk_speed   (info : in tae_wpt.event_context_ptr);
procedure trk_mod_trk_height (info : in tae_wpt.event_context_ptr);
procedure trk_mod_trk_mod_enter
                                (info : in tae_wpt.event_context_ptr);
procedure trk_mod_trk_plat_class
                                (info : in tae_wpt.event_context_ptr);
procedure filt_sel_canx_filter
                                (info : in tae_wpt.event_context_ptr);
procedure filt_sel_type_select
                                (info : in tae_wpt.event_context_ptr);
procedure filt_sel_filter_complete
                                (info : in tae_wpt.event_context_ptr);
procedure filt_sel_add_filter(info : in tae_wpt.event_context_ptr);
procedure rng_filt_rng_type   (info : in tae_wpt.event_context_ptr);
procedure rng_filt_rng_limit (info : in tae_wpt.event_context_ptr);
procedure rng_filt_hgt_limit (info : in tae_wpt.event_context_ptr);
procedure rng_filt_operator   (info : in tae_wpt.event_context_ptr);
procedure rng_filt_complete   (info : in tae_wpt.event_context_ptr);
procedure rng_filt_cancel     (info : in tae_wpt.event_context_ptr);
procedure cat_filt_cancel     (info : in tae_wpt.event_context_ptr);
procedure cat_filt_cat_type   (info : in tae_wpt.event_context_ptr);
procedure cat_filt_operator   (info : in tae_wpt.event_context_ptr);
procedure id_filt_id_type     (info : in tae_wpt.event_context_ptr);
procedure id_filt_cancel      (info : in tae_wpt.event_context_ptr);
procedure id_filt_Operator    (info : in tae_wpt.event_context_ptr);
procedure int_cept_method     (info : in tae_wpt.event_context_ptr);
procedure int_cept_time       (info : in tae_wpt.event_context_ptr);
procedure int_cept_speed      (info : in tae_wpt.event_context_ptr);
```

```
    procedure int_cept_compute    (info : in tae_wpt.event_context_ptr);
    procedure int_cept_cancel     (info : in tae_wpt.event_context_ptr);
-- END EVENT_HANDLERs
end menu_support;
-- Program Declarations
-------------------------------------------------------------------------

    use menu_support;
    use tae.tae_misc;
    use Events;
    use Graphic_Output;

    Tacplot_Buffer            : X_Lib.String_Pointer := new String'(" ");
    Tacplot_Symbol            : X_Lib.Keyboard.Key_Sym;
    Tacplot_Status            : Compose_Status_Record;

    Tacplot_X_Event           : Event := new Events.Event_Record;
    System_Info_X_Event       : Event := new Events.Event_Record;
    Track_Info_X_Event        : Event := new Events.Event_Record;
    Alerts_X_Event            : Event := new Events.Event_Record;
    Recommendations_X_Event   : Event := new Events.Event_Record;
    Intel_X_Event             : Event := new Events.Event_Record;
    System_Status_X_Event     : Event := new Events.Event_Record;

    theDisplay                : X_Windows.Display;
    user_ptr                  : tae_wpt.event_context_ptr;
    menu_info                 : tae_wpt.event_context_ptr;
    Sorry_info                : tae_wpt.event_context_ptr;
    trk_sel_info              : tae_wpt.event_context_ptr;
    sel_brg_info              : tae_wpt.event_context_ptr;
    glob_pos_info             : tae_wpt.event_context_ptr;
    confirm_info              : tae_wpt.event_context_ptr;
```

```
    alrt_sel_info           : tae_wpt.event_context_ptr;
    eqpt_sel_info           : tae_wpt.event_context_ptr;
    title_info              : tae_wpt.event_context_ptr;
    exit_ok_info            : tae_wpt.event_context_ptr;
    trk_mod_info            : tae_wpt.event_context_ptr;
    filt_sel_info           : tae_wpt.event_context_ptr;
    rng_filt_info           : tae_wpt.event_context_ptr;
    cat_filt_info           : tae_wpt.event_context_ptr;
    id_filt_info            : tae_wpt.event_context_ptr;
    int_cept_info           : tae_wpt.event_context_ptr;
    etype                   : wpt_eventtype;
    wptEvent                : tae_wpt.wpt_eventptr;

-------------------------------------------------------------------------
package body menu_support is
procedure initializePanels (file : in string) is
use tae.tae_co;
use tae.tae_misc;
tmp_info : tae_wpt.event_context_ptr;
     begin
-- do one Co_New and Co_ReadFile per resource file
tmp_info := new tae_wpt.event_context;
Co_New (0, tmp_info.collection);
-- could pass P_ABORT if you prefer
Co_ReadFile (tmp_info.collection, file, P_CONT);

       -- pair of Co_Finds for each panel in this resource file

        menu_info := new tae_wpt.event_context;
        menu_info.collection := tmp_info.collection;
        Co_Find (menu_info.collection, "menu_v", menu_info.view);
```

```
Co_Find (menu_info.collection, "menu_t", menu_info.target);

Sorry_info := new tae_wpt.event_context;
Sorry_info.collection := tmp_info.collection;
Co_Find (Sorry_info.collection, "Sorry_v", Sorry_info.view);
Co_Find (Sorry_info.collection, "Sorry_t", Sorry_info.target);

trk_sel_info := new tae_wpt.event_context;
trk_sel_info.collection := tmp_info.collection;
Co_Find (trk_sel_info.collection,
        "trk_sel_v", trk_sel_info.view);
Co_Find (trk_sel_info.collection,
        "trk_sel_t", trk_sel_info.target);

sel_brg_info := new tae_wpt.event_context;
sel_brg_info.collection := tmp_info.collection;
Co_Find (sel_brg_info.collection,
        "sel_brg_v", sel_brg_info.view);
Co_Find (sel_brg_info.collection,
        "sel_brg_t", sel_brg_info.target);
glob_pos_info := new tae_wpt.event_context;
glob_pos_info.collection := tmp_info.collection;
Co_Find (glob_pos_info.collection,
        "glob_pos_v", glob_pos_info.view);
Co_Find (glob_pos_info.collection,
        "glob_pos_t", glob_pos_info.target);
confirm_info := new tae_wpt.event_context;
confirm_info.collection := tmp_info.collection;
Co_Find (confirm_info.collection,
        "confirm_v", confirm_info.view);
Co_Find (confirm_info.collection,
```

```
        "confirm_t", confirm_info.target);
alrt_sel_info := new tae_wpt.event_context;
alrt_sel_info.collection := tmp_info.collection;
Co_Find (alrt_sel_info.collection,
        "alrt_sel_v", alrt_sel_info.view);
Co_Find (alrt_sel_info.collection,
        "alrt_sel_t", alrt_sel_info.target);
 eqpt_sel_info := new tae_wpt.event_context;
eqpt_sel_info.collection := tmp_info.collection;
Co_Find (eqpt_sel_info.collection,
        "eqpt_sel_v", eqpt_sel_info.view);
Co_Find (eqpt_sel_info.collection,
        "eqpt_sel_t", eqpt_sel_info.target);
title_info := new tae_wpt.event_context;
title_info.collection := tmp_info.collection;
Co_Find (title_info.collection, "title_v", title_info.view);
Co_Find (title_info.collection, "title_t", title_info.target);
 exit_ok_info := new tae_wpt.event_context;
exit_ok_info.collection := tmp_info.collection;
Co_Find (exit_ok_info.collection,
        "exit_ok_v", exit_ok_info.view);
Co_Find (exit_ok_info.collection,
        "exit_ok_t", exit_ok_info.target);
trk_mod_info := new tae_wpt.event_context;
trk_mod_info.collection := tmp_info.collection;
Co_Find (trk_mod_info.collection,
        "trk_mod_v", trk_mod_info.view);
Co_Find (trk_mod_info.collection,
        "trk_mod_t", trk_mod_info.target);
filt_sel_info := new tae_wpt.event_context;
filt_sel_info.collection := tmp_info.collection;
```

```
Co_Find (filt_sel_info.collection,
        "filt_sel_v", filt_sel_info.view);
Co_Find (filt_sel_info.collection,
        "filt_sel_t", filt_sel_info.target);
rng_filt_info := new tae_wpt.event_context;
rng_filt_info.collection := tmp_info.collection;
Co_Find (rng_filt_info.collection,
        "rng_filt_v", rng_filt_info.view);
Co_Find (rng_filt_info.collection,
        "rng_filt_t", rng_filt_info.target);
cat_filt_info := new tae_wpt.event_context;
cat_filt_info.collection := tmp_info.collection;
Co_Find (cat_filt_info.collection,
        "cat_filt_v", cat_filt_info.view);
Co_Find (cat_filt_info.collection,
        "cat_filt_t", cat_filt_info.target);
id_filt_info := new tae_wpt.event_context;
id_filt_info.collection := tmp_info.collection;
Co_Find (id_filt_info.collection,
        "id_filt_v", id_filt_info.view);
Co_Find (id_filt_info.collection,
        "id_filt_t", id_filt_info.target);
int_cept_info := new tae_wpt.event_context;
int_cept_info.collection := tmp_info.collection;
Co_Find (int_cept_info.collection,
        "int_cept_v", int_cept_info.view);
Co_Find (int_cept_info.collection,
        "int_cept_t", int_cept_info.target);
-- Since there can now be MULTIPLE INITIAL PANELS defined from
-- within the TAE WorkBench, call Wpt_NewPanel for each panel
-- defined to be an initial panel (but not usually all the panels
```

```
        -- which appear in the resource file).
         tae_wpt.Wpt_NewPanel ("", title_info.target, title_info.view,
            X_Windows.Null_Window, title_info, tae_wpt.WPT_DEFAULT,
            title_info.panel_id);
         tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
            X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
            menu_info.panel_id);
      end initializePanels;
---BEGIN EVENT_HANDLERs
procedure menu_track_button (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;
        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        OWNSHIP := GET_OWNSHIP_TRACK;
        if s_equal (value(1), "Add a New Track") then
            ADD_TRACK_FLAG := TRUE;
            tae_wpt.Wpt_NewPanel
                        ("", trk_sel_info.target, trk_sel_info.view,
              X_Windows.Null_Window, trk_sel_info, tae_wpt.WPT_DEFAULT,
               trk_sel_info.panel_id);

        elsif s_equal (value(1), "Delete the Selected Track") then
    DELETE_TRACK := GET_HOOKED_TRACK_NUMBER;
            if DELETE_TRACK /= 9999 then
            tae_wpt.Wpt_NewPanel
```

```
                        ("", confirm_info.target, confirm_info.view,
                X_Windows.Null_Window, confirm_info, tae_wpt.WPT_DEFAULT,
                 confirm_info.panel_id);
              else
                null;
              end if;
          elsif s_equal (value(1), "Modify the Selected Track") then
              MOD_TRACK_FLAG := TRUE;
              MOD_TRACK := GET_HOOKED_TRACK_NUMBER;
      if MOD_TRACK /= 9999 then
                 tae_wpt.Wpt_NewPanel
                        ("", trk_mod_info.target, trk_mod_info.view,
                   X_Windows.Null_Window,
                                  trk_mod_info, tae_wpt.WPT_DEFAULT,
                   trk_mod_info.panel_id);
              else
                 null;
              end if;
           end if;
      end menu_track_button;
  procedure menu_plots_menu (info : in tae_wpt.event_context_ptr) is
      value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
      count : taeint;
          begin
          tae_vm.Vm_Extract_Count (info.parm_ptr, count);
          if count <= 0 then
             null;
          else
             tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
          end if;
```

```
        if s_equal (value(1), "Maps and Grids") then null;
           tae_wpt.Wpt_NewPanel ("", Sorry_info.target, Sorry_info.view,
                X_Windows.Null_Window, Sorry_info, tae_wpt.WPT_DEFAULT,
                Sorry_info.panel_id);
        elsif s_equal (value(1), "Zones and Areas") then null;
           tae_wpt.Wpt_NewPanel ("", Sorry_info.target, Sorry_info.view,
                X_Windows.Null_Window, Sorry_info, tae_wpt.WPT_DEFAULT,
                Sorry_info.panel_id);
        elsif s_equal (value(1), "Sectors and Formations") then null;
           tae_wpt.Wpt_NewPanel ("", Sorry_info.target, Sorry_info.view,
                X_Windows.Null_Window, Sorry_info, tae_wpt.WPT_DEFAULT,
                Sorry_info.panel_id);
        end if;
    end menu_plots_menu;
procedure menu_alerts_button (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;
        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        if s_equal (value(1), "Visual Alerts Only") then
          SET_VISUAL_ALERTS;
        elsif s_equal (value(1), "Audio and Visual Alerts") then
          SET_AV_ALERTS;
        elsif s_equal (value(1), "Disable all Alerts") then
          DISABLE_ALERTS;
        end if;
```

```
   end menu_alerts_button;
procedure menu_filters_button (info : in tae_wpt.event_context_ptr) is
   value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
   count : taeint;
    begin
       tae_vm.Vm_Extract_Count (info.parm_ptr, count);
       if count <= 0 then
          null;
       else
          tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
       end if;
       if s_equal (value(1), "Disable all Display Filters") then
         INTEGRATION_SYSTEM.CLEAR_FILTER;
         SET_FILTER_STATUS ("OFF");
       elsif s_equal (value(1), "Activate Default Display Filters") then
         INTEGRATION_SYSTEM.CLEAR_FILTER;
         SET_FILTER_STATUS ("OFF");
       end if;
  end menu_filters_button;
procedure menu_defaults_button (info : in tae_wpt.event_context_ptr) is
   value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
   count : taeint;
       begin
       tae_vm.Vm_Extract_Count (info.parm_ptr, count);
       if count <= 0 then
          null;
       else
          tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
       end if;
        if s_equal (value(1), "Set System Filters") then null;
           tae_wpt.Wpt_NewPanel
```

```
                    ("", filt_sel_info.target, filt_sel_info.view,
            X_Windows.Null_Window,
                                filt_sel_info, tae_wpt.WPT_DEFAULT,
                                filt_sel_info.panel_id);
        elsif s_equal (value(1), "Set Alert Parameters") then null;
           tae_wpt.Wpt_NewPanel
                    ("", alrt_sel_info.target, alrt_sel_info.view,
           X_Windows.Null_Window,
                                alrt_sel_info, tae_wpt.WPT_DEFAULT,
                                alrt_sel_info.panel_id);
      elsif s_equal (value(1), "Set External System Inputs") then null;
           tae_wpt.Wpt_NewPanel
                    ("", eqpt_sel_info.target, eqpt_sel_info.view,
                     X_Windows.Null_Window, eqpt_sel_info,
                     tae_wpt.WPT_DEFAULT, eqpt_sel_info.panel_id);
      elsif s_equal (value(1), "Set Custom System Configurations") then
        null;
         tae_wpt.Wpt_NewPanel ("", Sorry_info.target, Sorry_info.view,
             X_Windows.Null_Window, Sorry_info, tae_wpt.WPT_DEFAULT,
             Sorry_info.panel_id);
      end if;
   end menu_defaults_button;
procedure menu_intel_button (info : in tae_wpt.event_context_ptr) is
   value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
   count : taeint;
      begin
      tae_vm.Vm_Extract_Count (info.parm_ptr, count);
      if count <= 0 then
         null;
      else
         tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
```

```
        end if;
        if s_equal (value(1), "Use Name for Intel Search") then null;
           tae_wpt.Wpt_NewPanel ("", Sorry_info.target, Sorry_info.view,
                X_Windows.Null_Window, Sorry_info, tae_wpt.WPT_DEFAULT,
                Sorry_info.panel_id);
        elsif s_equal (value(1),
          "Use Country and Platform Type for Intel Search") then null;
           tae_wpt.Wpt_NewPanel ("", Sorry_info.target, Sorry_info.view,
                X_Windows.Null_Window, Sorry_info, tae_wpt.WPT_DEFAULT,
                Sorry_info.panel_id);
        elsif s_equal (value(1),
          "Use Comments for Intel Search") then null;
           tae_wpt.Wpt_NewPanel ("", Sorry_info.target, Sorry_info.view,
                X_Windows.Null_Window, Sorry_info, tae_wpt.WPT_DEFAULT,
                Sorry_info.panel_id);
       elsif s_equal (value(1), "Edit Intelligence Database") then null;
           tae_wpt.Wpt_NewPanel ("", Sorry_info.target, Sorry_info.view,
                X_Windows.Null_Window, Sorry_info, tae_wpt.WPT_DEFAULT,
                Sorry_info.panel_id);
        end if;
    end menu_intel_button;
procedure menu_nav_button (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;
        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
```

```
        if s_equal (value(1), "Plot a Great Circle Path") then null;
           tae_wpt.Wpt_NewPanel ("", Sorry_info.target, Sorry_info.view,
                X_Windows.Null_Window, Sorry_info, tae_wpt.WPT_DEFAULT,
                Sorry_info.panel_id);
        elsif s_equal (value(1),
          "Override Navigation Equipment Inputs") then null;
           tae_wpt.Wpt_NewPanel ("", Sorry_info.target, Sorry_info.view,
                X_Windows.Null_Window, Sorry_info, tae_wpt.WPT_DEFAULT,
                Sorry_info.panel_id);
        elsif s_equal (value(1), "Plot an Intercept Course") then
          OWNSHIP := GET_OWNSHIP_TRACK;
          INTERCEPT_TRACK_NO := GET_HOOKED_TRACK_NUMBER;
            if INTERCEPT_TRACK_NO /= 9999 then
              INTERCEPT_TRACK := GET_HOOKED_TRACK_OBJECT;
              tae_wpt.Wpt_NewPanel
                        ("", int_cept_info.target, int_cept_info.view,
                         X_Windows.Null_Window, int_cept_info,
                         tae_wpt.WPT_DEFAULT, int_cept_info.panel_id);
            else null;
            end if;
         elsif s_equal (value(1), "Plot an Avoidance Course") then null;
           tae_wpt.Wpt_NewPanel ("", Sorry_info.target, Sorry_info.view,
                X_Windows.Null_Window, Sorry_info, tae_wpt.WPT_DEFAULT,
                Sorry_info.panel_id);
        end if;
    end menu_nav_button;
procedure menu_lists_button (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;
```

```
        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;

        if s_equal (value(1), "View a List") then null;
          tae_wpt.Wpt_NewPanel ("", Sorry_info.target, Sorry_info.view,
               X_Windows.Null_Window, Sorry_info, tae_wpt.WPT_DEFAULT,
               Sorry_info.panel_id);
        elsif s_equal (value(1), "Edit a List") then null;
          tae_wpt.Wpt_NewPanel ("", Sorry_info.target, Sorry_info.view,
               X_Windows.Null_Window, Sorry_info, tae_wpt.WPT_DEFAULT,
               Sorry_info.panel_id);
        end if;
    end menu_lists_button;

procedure menu_coding_button (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
```

```
        if s_equal (value(1), "Decode an ATP-1 Signal") then null;
          tae_wpt.Wpt_NewPanel ("", Sorry_info.target, Sorry_info.view,
              X_Windows.Null_Window, Sorry_info, tae_wpt.WPT_DEFAULT,
              Sorry_info.panel_id);
        elsif s_equal (value(1), "Encode an ATP-1 Signal") then null;
          tae_wpt.Wpt_NewPanel ("", Sorry_info.target, Sorry_info.view,
              X_Windows.Null_Window, Sorry_info, tae_wpt.WPT_DEFAULT,
              Sorry_info.panel_id);
        end if;
    end menu_coding_button;

procedure Sorry_OK_Button (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
           X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
           menu_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end Sorry_OK_Button;

procedure Sorry_OK_2 (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;
```

```
        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
           X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
           menu_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end Sorry_OK_2;

procedure trk_sel_NoName01 (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
           X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
           menu_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end trk_sel_NoName01;
```

```
procedure trk_sel_trk_sel_brg_rng
                                (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
       tae_wpt.Wpt_NewPanel ("", sel_brg_info.target, sel_brg_info.view,
           X_Windows.Null_Window, sel_brg_info, tae_wpt.WPT_DEFAULT,
           sel_brg_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end trk_sel_trk_sel_brg_rng;

procedure trk_sel_trk_sel_screen
                                (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        tae_wpt.Wpt_NewPanel ("", Sorry_info.target, Sorry_info.view,
```

```
            X_Windows.Null_Window, Sorry_info, tae_wpt.WPT_DEFAULT,
            Sorry_info.panel_id);
    end trk_sel_trk_sel_screen;

procedure trk_sel_trk_sel_cancel
                                (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
           X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
           menu_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end trk_sel_trk_sel_cancel;

procedure trk_sel_trk_sel_lat_lon
                                (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
```

```
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        tae_wpt.Wpt_NewPanel
          ("", glob_pos_info.target, glob_pos_info.view,
           X_Windows.Null_Window, glob_pos_info, tae_wpt.WPT_DEFAULT,
           glob_pos_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end trk_sel_trk_sel_lat_lon;

procedure sel_scrn_sel_scrn_canx
                                (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
       tae_wpt.Wpt_NewPanel ("", trk_sel_info.target, trk_sel_info.view,
           X_Windows.Null_Window, trk_sel_info, tae_wpt.WPT_DEFAULT,
           trk_sel_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end sel_scrn_sel_scrn_canx;

procedure sel_brg_tgt_bearing (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of taeint;
    count : taeint;
```

```
        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_IVAL (info.parm_ptr, 1, value(1));
           Target_Bearing := DEGREES_TO_RADIANS(FLOAT(value(1)));
        end if;
    end sel_brg_tgt_bearing;

procedure sel_brg_tgt_range (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of taeint;
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_IVAL (info.parm_ptr, 1, value(1));
           Target_Range := FLOAT(value(1));
        end if;
    end sel_brg_tgt_range;

procedure sel_brg_sel_brg_canx_1
                                (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
```

```
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
       tae_wpt.Wpt_NewPanel ("", trk_sel_info.target, trk_sel_info.view,
           X_Windows.Null_Window, trk_sel_info, tae_wpt.WPT_DEFAULT,
           trk_sel_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end sel_brg_sel_brg_canx_1;

procedure sel_brg_Enter_Data (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        TGT_BRG_RNG := RELATIVE_POSITION
                       (MAKE_POLAR_VECTOR_2
                       (TARGET_RANGE, TARGET_BEARING));
        TARGET_OBS := MAKE_GLOBAL_OBSERVATION
                      (OWNSHIP, TRK, TGT_BRG_RNG);
        INTEGRATION_SYSTEM.CREATE_TRACK (TARGET_OBS, Track_pkg.UNKNOWN);
        ADD_TRACK_FLAG := FALSE;
        tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
```

```
            X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
            menu_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end sel_brg_Enter_Data;


procedure sel_brg_sel_brg_canx (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
            null;
        else
            tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
       tae_wpt.Wpt_NewPanel ("", trk_sel_info.target, trk_sel_info.view,
            X_Windows.Null_Window, trk_sel_info, tae_wpt.WPT_DEFAULT,
            trk_sel_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end sel_brg_sel_brg_canx;


procedure glob_pos_long_deg (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of taeint;
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
```

```
            null;
         else
            tae_vm.Vm_Extract_IVAL (info.parm_ptr, 1, value(1));
            LONG_DEG:= NATURAL(VALUE(1));
         end if;
      end glob_pos_long_deg;


procedure glob_pos_long_min (info : in tae_wpt.event_context_ptr) is
      value : array (1..1) of taeint;
      count : taeint;

         begin
         tae_vm.Vm_Extract_Count (info.parm_ptr, count);
         if count <= 0 then
            null;
         else
            tae_vm.Vm_Extract_IVAL (info.parm_ptr, 1, value(1));
            LONG_MIN:= NATURAL(VALUE(1));
         end if;
      end glob_pos_long_min;

procedure glob_pos_long_sec (info : in tae_wpt.event_context_ptr) is
      value : array (1..1) of taeint;
      count : taeint;

         begin
         tae_vm.Vm_Extract_Count (info.parm_ptr, count);
         if count <= 0 then
            null;
         else
```

```
            tae_vm.Vm_Extract_IVAL (info.parm_ptr, 1, value(1));
            LONG_SEC:= NATURAL(VALUE(1));
        end if;
    end glob_pos_long_sec;

procedure glob_pos_long_dir (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
            null;
        else
            tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
            if s_equal (value(1),"E")then
              LONG_DIR := E;
            else
              LONG_DIR := W;
            end if;
        end if;
    end glob_pos_long_dir;

procedure glob_pos_lat_deg (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of taeint;
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
            null;
```

```
        else
           tae_vm.Vm_Extract_IVAL (info.parm_ptr, 1, value(1));
           LAT_DEG:= NATURAL(VALUE(1));
        end if;
    end glob_pos_lat_deg;

procedure glob_pos_lat_min (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of taeint;
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_IVAL (info.parm_ptr, 1, value(1));
           LAT_MIN:= NATURAL(VALUE(1));
        end if;
    end glob_pos_lat_min;

procedure glob_pos_lat_sec (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of taeint;
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_IVAL (info.parm_ptr, 1, value(1));
           LAT_SEC:= NATURAL(VALUE(1));
```

```
        end if;
    end glob_pos_lat_sec;

 procedure glob_pos_lat_dir (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
           if s_equal (value(1),"N")then
             LAT_DIR := N;
           else
             LAT_DIR := S;
           end if;
        end if;
    end glob_pos_lat_dir;

procedure glob_pos_spacer (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
```

```
        end if;
       tae_wpt.Wpt_NewPanel ("", trk_sel_info.target, trk_sel_info.view,
           X_Windows.Null_Window, trk_sel_info, tae_wpt.WPT_DEFAULT,
           trk_sel_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end glob_pos_spacer;

-- global position Enter Data button

procedure glob_pos_Spacer_1 (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;
    NEW_POSITION    : GLOBAL_POSITION;
    NEW_OBSERVATION : GLOBAL_OBSERVATION;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
      end if;
           if Mod_Track_Flag then
              NEW_POSITION := MAKE_GLOBAL_POSITION
                              (LAT_DIR, LAT_DEG, LAT_MIN, LAT_SEC,
                               LONG_DIR, LONG_DEG, LONG_MIN, LONG_SEC);

              INTEGRATION_SYSTEM.CHANGE_GLOBAL_POSITION
                                              (MOD_TRACK, NEW_POSITION);
              Mod_Track_Flag := False;
              Mod_Track :=9999;
```

```
            else
              NEW_OBSERVATION.POSITION := MAKE_GLOBAL_POSITION
                            (LAT_DIR, LAT_DEG, LAT_MIN, LAT_SEC,
                             LONG_DIR, LONG_DEG, LONG_MIN, LONG_SEC);
              NEW_OBSERVATION.OBSERVATION_TIME :=
                                                  ABSOLUTE_TIME_PKG.NOW;
              NEW_OBSERVATION.COURSE_AND_SPEED :=
                                 (MAKE_CARTESIAN_VECTOR_2 (0.0, 0.0));
              INTEGRATION_SYSTEM.CREATE_TRACK
                                 (NEW_OBSERVATION, TRACK_PKG.UNKNOWN);
              Add_Track_Flag := False;
            end if;

         tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
            X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
            menu_info.panel_id);
         tae_wpt.Wpt_PanelErase(info.panel_id);
      end glob_pos_Spacer_1;

procedure exit_ok_exit_continue (info : in tae_wpt.event_context_ptr) is
      value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
      count : taeint;

         begin
         tae_vm.Vm_Extract_Count (info.parm_ptr, count);
         if count <= 0 then
            null;
         else
            tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
         end if;
      end exit_ok_exit_continue;
```

```
procedure exit_ok_exit_cancel (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
           X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
           menu_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end exit_ok_exit_cancel;

procedure trk_mod_trk_identity (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;
    TRACK_ID: IDENTITY_TYPE;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
           if s_equal (value(1), "Friendly") then
             TRACK_ID := FRIENDLY;
```

```
            elsif s_equal (value(1), "Hostile") then
              TRACK_ID := HOSTILE;
            elsif s_equal (value(1), "Neutral") then
              TRACK_ID := NEUTRAL;
            elsif s_equal (value(1), "Unknown") then
              TRACK_ID := TRACK_PKG.UNKNOWN;
            end if;
           INTEGRATION_SYSTEM.SET_TRACK_IDENTITY (MOD_TRACK, TRACK_ID);
        end if;
    end trk_mod_trk_identity;

procedure trk_mod_trk_control (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;
    TRACK_CONTROL: CONTROL_TYPE;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
           if s_equal (value(1), "LINK") then
             TRACK_CONTROL := LINK;
           else
             TRACK_CONTROL := LOCAL;
           end if;
        end if;
        INTEGRATION_SYSTEM.SET_CONTROL (MOD_TRACK, TRACK_CONTROL);
    end trk_mod_trk_control;
```

```
procedure trk_mod_trk_point_type
                                (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
    end trk_mod_trk_point_type;

procedure trk_mod_trk_category (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;
    TRACK_CAT: TRACK_CATEGORY;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
          null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
           if s_equal (value(1), "Unknown") then
             TRACK_CAT := TRACK_PKG.UNKNOWN;
           elsif s_equal (value(1), "Surface_Platform") then
             TRACK_CAT := SURFACE_PLATFORM;
           elsif s_equal (value(1), "SubSurface_Platform") then
             TRACK_CAT := SUBSURFACE_PLATFORM;
```

```
            elsif s_equal (value(1), "Air_Platform") then
              TRACK_CAT := AIR_PLATFORM;
            elsif s_equal (value(1), "Man_In_Water") then
              TRACK_CAT := MAN_IN_WATER;
            elsif s_equal (value(1), "Special_Point") then
              TRACK_CAT := SPECIAL_POINT;
            end if;
         end if;
         INTEGRATION_SYSTEM.CHANGE_TRACK_CATEGORY (MOD_TRACK, TRACK_CAT);
    end trk_mod_trk_category;

procedure trk_mod_trk_position (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        tae_wpt.Wpt_NewPanel
          ("", glob_pos_info.target, glob_pos_info.view,
           X_Windows.Null_Window, glob_pos_info, tae_wpt.WPT_DEFAULT,
           glob_pos_info.panel_id);
        tae_wpt.Wpt_PanelErase(trk_mod_info.panel_id);

    end trk_mod_trk_position;
```

```
procedure trk_mod_trk_amp_info (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..56);
    count : taeint;
    TRACK_AMP : STRING (1..56);
        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
           TRACK_AMP := VALUE(1);
        end if;
        INTEGRATION_SYSTEM.SET_AMPL_INFO (MOD_TRACK, TRACK_AMP);
    end trk_mod_trk_amp_info;

procedure trk_mod_trk_name (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..32);
    count : taeint;
    TRACK_NAME : STRING (1..32);

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
           TRACK_NAME := VALUE(1);
        end if;
        INTEGRATION_SYSTEM.SET_VESSEL_NAME (MOD_TRACK, TRACK_NAME);
    end trk_mod_trk_name;
```

```
procedure trk_mod_trk_course (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of taeint;
    count : taeint;
    TARGET_CSE : ANGLE_PKG.ANGLE;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_IVAL (info.parm_ptr, 1, value(1));
           TARGET_CSE := DEGREES_TO_RADIANS(FLOAT(VALUE(1)));
        end if;
       INTEGRATION_SYSTEM.CHANGE_COURSE (MOD_TRACK, TARGET_CSE);
    end trk_mod_trk_course;

procedure trk_mod_trk_speed (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of taeint;
    count : taeint;
    TARGET_SPD : SPEED;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_IVAL (info.parm_ptr, 1, value(1));
           TARGET_SPD := MAKE_SPEED(FLOAT(VALUE(1)));
        end if;
        INTEGRATION_SYSTEM.CHANGE_SPEED (MOD_TRACK, TARGET_SPD);
    end trk_mod_trk_speed;
```

```
procedure trk_mod_trk_height (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of taeint;
    count : taeint;
    TARGET_HEIGHT : DISTANCE;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_IVAL (info.parm_ptr, 1, value(1));
           TARGET_HEIGHT := (FLOAT(VALUE(1))/3.0);
        end if;
      INTEGRATION_SYSTEM.SET_ALTITUDE (MOD_TRACK, TARGET_HEIGHT);
    end trk_mod_trk_height;

procedure trk_mod_trk_mod_enter (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        MOD_TRACK_FLAG := FALSE;
        MOD_TRACK := 9999;
        tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
```

```
            X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
            menu_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end trk_mod_trk_mod_enter;

procedure trk_mod_trk_plat_class
                                (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..32);
    count : taeint;
    TRACK_CLASS : STRING (1..32);

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
           TRACK_CLASS := VALUE(1);
        end if;
        INTEGRATION_SYSTEM.SET_PLATFORM_CLASS (MOD_TRACK, TRACK_CLASS);
    end trk_mod_trk_plat_class;

procedure confirm_conf_continue (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;
        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
        INTEGRATION_SYSTEM.DELETE_TRACK_AND_SEND_TO_HISTORY
                                                    (DELETE_TRACK);
        MOD_TRACK_FLAG := FALSE;
```

```
        MOD_TRACK := 9999;
        end if;
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end confirm_conf_continue;

procedure confirm_conf_cancel (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
          null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        MOD_TRACK_FLAG := FALSE;
        MOD_TRACK := 9999;
        tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
           X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
           menu_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end confirm_conf_cancel;

procedure alrt_sel_cpa_alrt_range
                                (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of taeint;
    count : taeint;
      begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
```

```
            null;
         else
            tae_vm.Vm_Extract_IVAL (info.parm_ptr, 1, value(1));
         end if;
         Alert_Range := float(value(1));
       end alrt_sel_cpa_alrt_range;

procedure alrt_sel_alrt_canx_1 (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
           X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
           menu_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end alrt_sel_alrt_canx_1;

procedure alrt_sel_enter_data (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
```

```
            text_io.put_line ("none");
         else
            tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
         end if;
         SET_CPA_ALERT_RANGE (Alert_Range);
         tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
            X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
            menu_info.panel_id);
         tae_wpt.Wpt_PanelErase(info.panel_id);
      end alrt_sel_enter_data;

   procedure alrt_sel_alrt_canx_2 (info : in tae_wpt.event_context_ptr) is
      value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
      count : taeint;

         begin
         tae_vm.Vm_Extract_Count (info.parm_ptr, count);
         if count <= 0 then
            null;
         else
            tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
         end if;
         tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
            X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
            menu_info.panel_id);
         tae_wpt.Wpt_PanelErase(info.panel_id);
      end alrt_sel_alrt_canx_2;

   procedure eqpt_sel_gps_sel (info : in tae_wpt.event_context_ptr) is
      value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
      count : taeint;
```

```
        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
           if not GPS_SYS then
             GPS_SYS:= True;
           else
             GPS_SYS := False;
           end if;
        end if;
    end eqpt_sel_gps_sel;

procedure eqpt_sel_radar_sel (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
           if not RADAR_SYS then
             RADAR_SYS:= True;
           else
             RADAR_SYS := False;
           end if;
```

```
        end if;
    end eqpt_sel_radar_sel;

procedure eqpt_sel_link_sel (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
           if not LINK_SYS then
             LINK_SYS := True;
           else
             LINK_SYS := False;
           end if;
         end if;
    end eqpt_sel_link_sel;

procedure eqpt_sel_gyro_sel (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
```

```
            if not GYRO_SYS then
              GYRO_SYS := True;
            else
              GYRO_SYS := False;
            end if;
        end if;
    end eqpt_sel_gyro_sel;

procedure eqpt_sel_fath_sel (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
           if not FATH_SYS then
             FATH_SYS := True;
           else
             FATH_SYS := False;
           end if;

        end if;
    end eqpt_sel_fath_sel;

procedure eqpt_sel_pit_sel (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;
```

```
        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
           if not PIT_SYS then
             PIT_SYS := True;
           else
             PIT_SYS := False;
           end if;
        end if;
    end eqpt_sel_pit_sel;

procedure eqpt_sel_enter_data (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        INTEGRATION_SYSTEM.GET_SENSOR_STATUS ( LINK, EQPT_STATUS );
        if EQPT_STATUS  = DOWN and  LINK_SYS = TRUE then
            INTEGRATION_SYSTEM.SET_SENSOR_STATUS ( LINK, UP );
        elsif EQPT_STATUS  = UP and  LINK_SYS = FALSE then
            INTEGRATION_SYSTEM.SET_SENSOR_STATUS ( LINK, DOWN );
        end if;
```

```
INTEGRATION_SYSTEM.GET_SENSOR_STATUS ( GPS, EQPT_STATUS );
if EQPT_STATUS  = DOWN and  GPS_SYS = TRUE then
  INTEGRATION_SYSTEM.SET_SENSOR_STATUS ( GPS, UP );
elsif EQPT_STATUS  = UP and  GPS_SYS = FALSE then
  INTEGRATION_SYSTEM.SET_SENSOR_STATUS ( GPS, DOWN );
end if;
INTEGRATION_SYSTEM.GET_SENSOR_STATUS ( RADAR, EQPT_STATUS );
if EQPT_STATUS  = DOWN and  RADAR_SYS = TRUE then
  INTEGRATION_SYSTEM.SET_SENSOR_STATUS ( RADAR, UP );
elsif EQPT_STATUS  = UP and  RADAR_SYS = FALSE then
  INTEGRATION_SYSTEM.SET_SENSOR_STATUS ( RADAR, DOWN );
end if;
INTEGRATION_SYSTEM.GET_SENSOR_STATUS ( PITSWORD, EQPT_STATUS );
if EQPT_STATUS  = DOWN and  PIT_SYS = TRUE then
  INTEGRATION_SYSTEM.SET_SENSOR_STATUS ( PITSWORD, UP );
elsif EQPT_STATUS  = UP and  PIT_SYS = FALSE then
  INTEGRATION_SYSTEM.SET_SENSOR_STATUS ( PITSWORD, DOWN );
end if;
INTEGRATION_SYSTEM.GET_SENSOR_STATUS ( GYRO, EQPT_STATUS );
if EQPT_STATUS  = DOWN and  GYRO_SYS = TRUE then
  INTEGRATION_SYSTEM.SET_SENSOR_STATUS ( GYRO, UP );
elsif EQPT_STATUS  = UP and  GYRO_SYS = FALSE then
  INTEGRATION_SYSTEM.SET_SENSOR_STATUS ( GYRO, DOWN );
end if;
INTEGRATION_SYSTEM.GET_SENSOR_STATUS ( FATHOMETER, EQPT_STATUS );
if EQPT_STATUS  = DOWN and  FATH_SYS = TRUE then
  INTEGRATION_SYSTEM.SET_SENSOR_STATUS ( FATHOMETER, UP );
elsif EQPT_STATUS  = UP and  FATH_SYS = FALSE then
  INTEGRATION_SYSTEM.SET_SENSOR_STATUS ( FATHOMETER, DOWN );
end if;
TEST_SYSTEM_STATUS;
```

```
        tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
           X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
           menu_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end eqpt_sel_enter_data;

procedure title_exit_system (info : in tae_wpt.event_context_ptr) is
  count : taeint;

  begin
    tae_vm.Vm_Extract_Count (info.parm_ptr, count);
      if count <= 0 then
        null;
      end if;
      tae_wpt.Wpt_NewPanel ("", exit_ok_info.target, exit_ok_info.view,
           X_Windows.Null_Window, exit_ok_info, tae_wpt.WPT_DEFAULT,
           exit_ok_info.panel_id);
  end title_exit_system;

procedure filt_sel_canx_filter (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
```

```
            X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
            menu_info.panel_id);
         tae_wpt.Wpt_PanelErase(info.panel_id);
   end filt_sel_canx_filter;

procedure filt_sel_type_select (info : in tae_wpt.event_context_ptr) is
   value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
   count : taeint;

      begin
      tae_vm.Vm_Extract_Count (info.parm_ptr, count);
      if count <= 0 then
         null;
      else
         tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
      end if;
      if s_equal (value(1), "Based on Range/Height") then null;
          FC:= DISTANCE_FILTER;
          tae_wpt.Wpt_NewPanel
           ("", rng_filt_info.target, rng_filt_info.view,
           X_Windows.Null_Window, rng_filt_info, tae_wpt.WPT_DEFAULT,
             rng_filt_info.panel_id);
          tae_wpt.Wpt_PanelErase(info.panel_id);
      elsif s_equal (value(1), "Based on Track Category") then null;
          tae_wpt.Wpt_NewPanel
           ("", cat_filt_info.target, cat_filt_info.view,
           X_Windows.Null_Window, cat_filt_info, tae_wpt.WPT_DEFAULT,
             cat_filt_info.panel_id);
          tae_wpt.Wpt_PanelErase(info.panel_id);
      elsif s_equal (value(1), "Based on Platform Identity") then null;
          tae_wpt.Wpt_NewPanel
```

```
            ("", id_filt_info.target, id_filt_info.view,
            X_Windows.Null_Window, id_filt_info, tae_wpt.WPT_DEFAULT,
             id_filt_info.panel_id);
          tae_wpt.Wpt_PanelErase(info.panel_id);
        end if;
    end filt_sel_type_select;

procedure filt_sel_add_filter (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        INTEGRATION_SYSTEM.ADD_AND_FILTER_TO_FILTER;
        SET_FILTER_STATUS ("ON");
    end filt_sel_add_filter;

  procedure filt_sel_filter_complete
                              (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
```

```
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        INTEGRATION_SYSTEM.WRITE_FILTER;
        tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
           X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
           menu_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end filt_sel_filter_complete;

procedure rng_filt_rng_type (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
           if s_equal (value(1), "Height") then
             DA := ALTITUDE;
           else
             DA := RANGE_FROM_REFERENCE_TRACK;
           end if;
        end if;
    end rng_filt_rng_type;

 procedure rng_filt_rng_limit (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of taeint;
    count : taeint;
```

```
        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_IVAL (info.parm_ptr, 1, value(1));
           RG := FLOAT(VALUE(1));
        end if;
    end rng_filt_rng_limit;

procedure rng_filt_hgt_limit (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of taeint;
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
          null;
        else
           tae_vm.Vm_Extract_IVAL (info.parm_ptr, 1, value(1));
           RG:= FLOAT(VALUE(1)/3);
        end if;
    end rng_filt_hgt_limit;

 procedure rng_filt_operator (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
```

```
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
           if s_equal (value(1), ">=") then
             RID := GREATER_OR_EQUAL;
           else
             RID := LESS_OR_EQUAL;
           end if;
        end if;
    end rng_filt_operator;

procedure rng_filt_complete (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
     INTEGRATION_SYSTEM.MAKE_DISTANCE_ATOMIC_FILTER (DA, RG, 0, RID);
        tae_wpt.Wpt_NewPanel
          ("", filt_sel_info.target, filt_sel_info.view,
           X_Windows.Null_Window, filt_sel_info, tae_wpt.WPT_DEFAULT,
           filt_sel_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end rng_filt_complete;
```

```
procedure rng_filt_cancel (info : in tae_wpt.event_context_ptr) is
   value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
   count : taeint;

      begin
      tae_vm.Vm_Extract_Count (info.parm_ptr, count);
      if count <= 0 then
         null;
      else
         tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
      end if;
      tae_wpt.Wpt_NewPanel
        ("", filt_sel_info.target, filt_sel_info.view,
         X_Windows.Null_Window, filt_sel_info, tae_wpt.WPT_DEFAULT,
         filt_sel_info.panel_id);
      tae_wpt.Wpt_PanelErase(info.panel_id);
   end rng_filt_cancel;

procedure cat_filt_cancel (info : in tae_wpt.event_context_ptr) is
   value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
   count : taeint;

      begin
      tae_vm.Vm_Extract_Count (info.parm_ptr, count);
      if count <= 0 then
         null;
      else
         tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
      end if;
      tae_wpt.Wpt_NewPanel
         ("", filt_sel_info.target, filt_sel_info.view,
```

```
            X_Windows.Null_Window, filt_sel_info, tae_wpt.WPT_DEFAULT,
            filt_sel_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end cat_filt_cancel;

procedure cat_filt_cat_type (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        if s_equal (value(1), "Unknown") then
          T_CAT := TRACK_PKG.UNKNOWN;
        elsif s_equal (value(1), "Surface Platform") then
          T_CAT := SURFACE_PLATFORM;
        elsif s_equal (value(1), "Subsurface Platform") then
          T_CAT := SUBSURFACE_PLATFORM;
        elsif s_equal (value(1), "Air Platform") then
          T_CAT := AIR_PLATFORM;
        elsif s_equal (value(1), "Special Point") then
          T_CAT := SPECIAL_POINT;
        elsif s_equal (value(1), "Man in Water") then
          T_CAT := MAN_IN_WATER;
        end if;
    end cat_filt_cat_type;
```

```
procedure cat_filt_operator (info : in tae_wpt.event_context_ptr) is
  value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
  count : taeint;

      begin
      tae_vm.Vm_Extract_Count (info.parm_ptr, count);
      if count <= 0 then
         null;
      else
         tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
      end if;
      if s_equal (value(1), "Equal") then
        EQ := EQUAL;
        INTEGRATION_SYSTEM.MAKE_TRACK_CATEGORY_ATOMIC_FILTER
                                                        (T_CAT, EQ);
          tae_wpt.Wpt_NewPanel
           ("", filt_sel_info.target, filt_sel_info.view,
           X_Windows.Null_Window, filt_sel_info, tae_wpt.WPT_DEFAULT,
             filt_sel_info.panel_id);
          tae_wpt.Wpt_PanelErase(info.panel_id);
      elsif s_equal (value(1), "Not Equal") then
        EQ := NOT_EQUAL;
        INTEGRATION_SYSTEM.MAKE_TRACK_CATEGORY_ATOMIC_FILTER
                                                        (T_CAT, EQ);
          tae_wpt.Wpt_NewPanel
           ("", filt_sel_info.target, filt_sel_info.view,
           X_Windows.Null_Window, filt_sel_info, tae_wpt.WPT_DEFAULT,
             filt_sel_info.panel_id);
          tae_wpt.Wpt_PanelErase(info.panel_id);
      end if;
  end cat_filt_operator;
```

```
 procedure id_filt_id_type (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        if s_equal (value(1), "Unknown") then
          PID := TRACK_PKG.UNKNOWN;
        elsif s_equal (value(1), "Friendly") then
          PID := FRIENDLY;
        elsif s_equal (value(1), "Hostile") then
          PID := HOSTILE;
        elsif s_equal (value(1), "Neutral") then
          PID := NEUTRAL;
        end if;
    end id_filt_id_type;

procedure id_filt_cancel (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
```

```
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        tae_wpt.Wpt_NewPanel
                        ("", filt_sel_info.target, filt_sel_info.view,
           X_Windows.Null_Window, filt_sel_info, tae_wpt.WPT_DEFAULT,
           filt_sel_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end id_filt_cancel;

procedure id_filt_Operator (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
        end if;
        if s_equal (value(1), "Equal") then
            EQ := EQUAL;
            INTEGRATION_SYSTEM.MAKE_PLATFORM_IDENTITY_ATOMIC_FILTER
                                                          (PID, EQ);
            tae_wpt.Wpt_NewPanel
             ("", filt_sel_info.target, filt_sel_info.view,
             X_Windows.Null_Window, filt_sel_info, tae_wpt.WPT_DEFAULT,
               filt_sel_info.panel_id);
            tae_wpt.Wpt_PanelErase(info.panel_id);
        elsif s_equal (value(1), "Not Equal") then
```

```
            EQ := NOT_EQUAL;
            INTEGRATION_SYSTEM.MAKE_PLATFORM_IDENTITY_ATOMIC_FILTER
                                                              (PID, EQ);
            tae_wpt.Wpt_NewPanel
             ("", filt_sel_info.target, filt_sel_info.view,
             X_Windows.Null_Window, filt_sel_info, tae_wpt.WPT_DEFAULT,
               filt_sel_info.panel_id);
            tae_wpt.Wpt_PanelErase(info.panel_id);
         end if;
      end id_filt_Operator;

   procedure int_cept_method (info : in tae_wpt.event_context_ptr) is
      value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
      count : taeint;

         begin
         tae_vm.Vm_Extract_Count (info.parm_ptr, count);
         if count <= 0 then
            null;
         else
            tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));

            if s_equal (value(1), "Time to Station") then
               INTERCEPT_BY_SPEED := FALSE;
               INTERCEPT_BY_TIME  := TRUE;
            elsif s_equal (value(1), "Speed to Station") then
               INTERCEPT_BY_TIME  := FALSE;
               INTERCEPT_BY_SPEED := TRUE;
            end if;
         end if;
      end int_cept_method;
```

```
   procedure int_cept_time (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of taeint;
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_IVAL (info.parm_ptr, 1, value(1));
           STATION_TIME := NATURAL(VALUE(1));
        end if;
    end int_cept_time;

procedure int_cept_speed (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of taeint;
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
           tae_vm.Vm_Extract_IVAL (info.parm_ptr, 1, value(1));
           STATION_SPEED := FLOAT(VALUE(1));
        end if;
    end int_cept_speed;

 procedure int_cept_compute (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
```

```
count : taeint;
INTERCEPT_RECORD    : INTERCEPT_TYPE;
INTERCEPT_SPEED     : SPEED;
INTERCEPT_TIME      : ABSOLUTE_TIME;
INTERCEPT_COURSE    : INTEGER;
INTERCEPT_VEL       : INTEGER;
INTERCEPT_HOURS     : NATURAL;
INTERCEPT_MINS      : NATURAL;

    begin
    tae_vm.Vm_Extract_Count (info.parm_ptr, count);
    if count <= 0 then
       null;
    else
       tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
    end if;
    INTERCEPT_SPEED := MAKE_SPEED (STATION_SPEED);
   INTERCEPT_TIME  := NOW + RELATIVE_TIME(FLOAT(STATION_TIME * 60));

    if INTERCEPT_BY_SPEED then
      INTERCEPT_RECORD := INTERCEPT(INTERCEPT_TRACK, OWNSHIP,
                          RELATIVE_POSITION
                         (MAKE_CARTESIAN_VECTOR_2 (0.0, 0.0)),
                          INTERCEPT_SPEED);
    else
      INTERCEPT_RECORD := INTERCEPT
                         (INTERCEPT_TRACK, OWNSHIP, RELATIVE_POSITION
                          (MAKE_CARTESIAN_VECTOR_2 (0.0, 0.0)),
                          INTERCEPT_TIME);
    end if;
    INTERCEPT_BY_SPEED := FALSE;
```

```
        INTERCEPT_BY_TIME   := TRUE;
        INTERCEPT_COURSE    := INTEGER(RADIANS_TO_DEGREES(COURSE
                               (INTERCEPT_RECORD.INTERCEPT_CRS_AND_SPD)));
        INTERCEPT_VEL       := INTEGER(SPEED_IN_KNOTS(SPD
                               (INTERCEPT_RECORD.INTERCEPT_CRS_AND_SPD)));
        INTERCEPT_HOURS     := HOURS(TIME_OF_DAY
                               (INTERCEPT_RECORD.TIME_TO_INTERCEPT));
        INTERCEPT_MINS      := MINUTES(TIME_OF_DAY
                               (INTERCEPT_RECORD.TIME_TO_INTERCEPT));
        WRITE_INTERCEPT_RECOMMENDATION
                                    (NATURAL'IMAGE(INTERCEPT_TRACK_NO),
                                     INTEGER'IMAGE(INTERCEPT_COURSE),
                                     INTEGER'IMAGE(INTERCEPT_VEL),
                                     NATURAL'IMAGE(INTERCEPT_HOURS),
                                     NATURAL'IMAGE(INTERCEPT_MINS));

        tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
           X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
           menu_info.panel_id);
        tae_wpt.Wpt_PanelErase(info.panel_id);
    end int_cept_compute;

procedure int_cept_cancel (info : in tae_wpt.event_context_ptr) is
    value : array (1..1) of string (1..tae_taeconf.STRINGSIZE);
    count : taeint;

        begin
        tae_vm.Vm_Extract_Count (info.parm_ptr, count);
        if count <= 0 then
           null;
        else
```

```
            tae_vm.Vm_Extract_SVAL (info.parm_ptr, 1, value(1));
         end if;
         tae_wpt.Wpt_NewPanel ("", menu_info.target, menu_info.view,
            X_Windows.Null_Window, menu_info, tae_wpt.WPT_DEFAULT,
            menu_info.panel_id);
         tae_wpt.Wpt_PanelErase(info.panel_id);
      end int_cept_cancel;
   end menu_support;

   begin

      f_force_lower (FALSE);     -- permit upper/lowercase file names
      tae_wpt.Wpt_Init ("",theDisplay);
      tae_wpt.Wpt_NewEvent (wptEvent);

      initializePanels ("menu.res");    -- single call
      Open_Tacplot_Display;
      Open_Track_Info_Display;
      Open_Alerts_Display;
      Open_Recommendations_Display;
      Open_Intel_Display;
      Open_System_Status_Display;



      OWN_OBS.OBSERVATION_TIME := NOW;
      OWN_OBS.POSITION := MAKE_GLOBAL_POSITION
                                          (N, 15, 44, 0, E, 65, 23, 0);
      OWN_CRS := DEGREES_TO_RADIANS (90.0);
      OWN_SPD := MAKE_SPEED (20.0);
      OWN_OBS.COURSE_AND_SPEED := MAKE_VELOCITY ( OWN_SPD, OWN_CRS );
      INTEGRATION_SYSTEM.CREATE_TRACK ( OWN_OBS, SURFACE_PLATFORM );
```

```
OWN_OBS.OBSERVATION_TIME := NOW;
OWN_OBS.POSITION := MAKE_GLOBAL_POSITION
                                        (N, 15, 46, 0, E, 65, 20, 0);
OWN_CRS := DEGREES_TO_RADIANS (45.0);
OWN_SPD := MAKE_SPEED (600.0);
OWN_OBS.COURSE_AND_SPEED := MAKE_VELOCITY ( OWN_SPD, OWN_CRS );
INTEGRATION_SYSTEM.CREATE_TRACK ( OWN_OBS, AIR_PLATFORM );

OWN_OBS.OBSERVATION_TIME := NOW;
OWN_OBS.POSITION := MAKE_GLOBAL_POSITION
                                        (N, 15, 40, 0, E, 65, 26, 0);
OWN_CRS := DEGREES_TO_RADIANS (235.0);
OWN_SPD := MAKE_SPEED (20.0);
OWN_OBS.COURSE_AND_SPEED := MAKE_VELOCITY ( OWN_SPD, OWN_CRS );
INTEGRATION_SYSTEM.CREATE_TRACK ( OWN_OBS, SURFACE_PLATFORM );

OWN_OBS.OBSERVATION_TIME := NOW;
OWN_OBS.POSITION := MAKE_GLOBAL_POSITION
                                        (N, 15, 35, 0, E, 65, 17, 0);
OWN_CRS := DEGREES_TO_RADIANS (170.0);
OWN_SPD := MAKE_SPEED (20.0);
OWN_OBS.COURSE_AND_SPEED := MAKE_VELOCITY ( OWN_SPD, OWN_CRS );
INTEGRATION_SYSTEM.CREATE_TRACK ( OWN_OBS, SURFACE_PLATFORM );

INTEGRATION_SYSTEM.FILL_TACPLOT;
DISPLAY_TACPLOT;
```

```
loop
   if X_Pending (Tacplot_Display) > 0 then
          X_Lib.Events.X_Next_Event
                                  (Tacplot_Display, Tacplot_X_Event);
      case Tacplot_X_Event.Kind is
         when Expose     =>
           if (Tacplot_X_Event.Expose_Notify.Count = 0) then
           INTEGRATION_SYSTEM.FILL_TACPLOT;
           DISPLAY_TACPLOT;
           end if;
         when Button_Press    =>
           Process_Hook_Position (Tacplot_X_Event.Button.X,
                                  Tacplot_X_Event.Button.Y);
           INTEGRATION_SYSTEM.FILL_TACPLOT;
           DISPLAY_TACPLOT;
         when others => null;
       end case;
     end if;

 if X_Pending (Track_Info_Display) > 0 then
  X_Lib.Events.X_Next_Event
                          (Track_Info_Display, Track_Info_X_Event);
   case Track_Info_X_Event.Kind is
     when others => null;
   end case;
 end if;

 if X_Pending (Alerts_Display) > 0 then
  X_Lib.Events.X_Next_Event (Alerts_Display, Alerts_X_Event);
   case Alerts_X_Event.Kind is
     when others => null;
```

```
        end case;
      end if;

      if X_Pending (Recommendations_Display) > 0 then
       X_Lib.Events.X_Next_Event (Recommendations_Display,
                                  Recommendations_X_Event);
        case Recommendations_X_Event.Kind is
          when button_press => X_Clear_Window
              (Recommendations_Display, Recommendations_Window);

          when others => null;
        end case;
      end if;

      if X_Pending (Intel_Display) > 0 then
       X_Lib.Events.X_Next_Event (Intel_Display, Intel_X_Event);
        case Intel_X_Event.Kind is
          when others => null;
        end case;
      end if;


      if X_Pending (System_Status_Display) > 0 then
       X_Lib.Events.X_Next_Event (System_Status_Display,
System_Status_X_Event);
        case System_Status_X_Event.Kind is
          when button_press => Test_System_Status;
          when expose => Test_System_Status;
          when others => null;
        end case;
      end if;
```

```
if TAE_WPT.WPT_PENDING = TRUE then

   tae_wpt.Wpt_NextEvent (wptEvent, etype);

  case etype is
   when wpt_eventtype'first .. -1 => null;
         -- iterate loop on Wpt_NextEvent error

   when tae_wpt.WPT_PARM_EVENT =>
     tae_wpt.Wpt_Extract_Context (wptEvent, user_ptr);
     tae_wpt.Wpt_Extract_Parm (wptEvent, user_ptr.parm_name);
     tae_wpt.Wpt_Extract_Data (wptEvent, user_ptr.datavm_ptr);
     tae_vm.Vm_Find (user_ptr.datavm_ptr, user_ptr.parm_name,
                              user_ptr.parm_ptr);


-- MENU PANEL EVENTS

                  if tae_wpt."=" (user_ptr, menu_info)  then
                  if s_equal ("track_button", user_ptr.parm_name) then
                       menu_track_button (user_ptr);
                   elsif s_equal
                            ("plots_menu", user_ptr.parm_name) then
                       menu_plots_menu (user_ptr);
                   elsif s_equal
                        ("alerts_button", user_ptr.parm_name) then
                       menu_alerts_button (user_ptr);
                   elsif s_equal
                       ("filters_button", user_ptr.parm_name) then
                       menu_filters_button (user_ptr);
                   elsif s_equal
```

```
            ("defaults_button", user_ptr.parm_name) then
            menu_defaults_button (user_ptr);
        elsif s_equal
              ("intel_button", user_ptr.parm_name) then
            menu_intel_button (user_ptr);
        elsif s_equal
              ("nav_button", user_ptr.parm_name) then
            menu_nav_button (user_ptr);
        elsif s_equal
              ("lists_button", user_ptr.parm_name) then
            menu_lists_button (user_ptr);
        elsif s_equal
             ("coding_button", user_ptr.parm_name) then
            menu_coding_button (user_ptr);
        end if;

-- SORRY PANEL EVENTS

    elsif tae_wpt."=" (user_ptr, Sorry_info)  then
       if s_equal ("OK_Button", user_ptr.parm_name) then
            Sorry_OK_Button (user_ptr);
       elsif s_equal ("OK_2", user_ptr.parm_name) then
            Sorry_OK_2 (user_ptr);
       end if;     -- END panel Sorry

-- TRACK SELECT PANEL EVENTS

    elsif tae_wpt."=" (user_ptr, trk_sel_info)  then
        if s_equal ("NoName01", user_ptr.parm_name) then
            trk_sel_NoName01 (user_ptr);
        elsif s_equal
```

```
                ("trk_sel_brg_rng", user_ptr.parm_name) then
                trk_sel_trk_sel_brg_rng (user_ptr);
            elsif s_equal
                ("trk_sel_screen", user_ptr.parm_name) then
                trk_sel_trk_sel_screen (user_ptr);
            elsif s_equal
                ("trk_sel_cancel", user_ptr.parm_name) then
                trk_sel_trk_sel_cancel (user_ptr);
            elsif s_equal
                ("trk_sel_lat_lon", user_ptr.parm_name) then
                trk_sel_trk_sel_lat_lon (user_ptr);
            end if;     -- END panel trk_sel

-- INPUT BEARING AND RANGE PANEL EVENTS

        elsif tae_wpt."=" (user_ptr, sel_brg_info)  then
            if s_equal
                ("tgt_bearing", user_ptr.parm_name) then
                sel_brg_tgt_bearing (user_ptr);
            elsif s_equal
                ("tgt_range", user_ptr.parm_name) then
                sel_brg_tgt_range (user_ptr);
            elsif s_equal
                ("sel_brg_canx_1", user_ptr.parm_name) then
                sel_brg_sel_brg_canx_1 (user_ptr);
            elsif s_equal
                ("Enter_Data", user_ptr.parm_name) then
                sel_brg_Enter_Data (user_ptr);
            elsif s_equal
                ("sel_brg_canx", user_ptr.parm_name) then
                sel_brg_sel_brg_canx (user_ptr);
```

```
            end if;     -- END panel sel_brg

-- ENTER GLOBAL POSITION PANEL EVENTS

        elsif tae_wpt."=" (user_ptr, glob_pos_info)  then
            if s_equal ("lat_min", user_ptr.parm_name) then
                glob_pos_lat_min (user_ptr);
            elsif s_equal
                      ("lat_sec", user_ptr.parm_name) then
                glob_pos_lat_sec (user_ptr);
            elsif s_equal
                     ("long_deg", user_ptr.parm_name) then
                glob_pos_long_deg (user_ptr);
            elsif s_equal
                     ("long_min", user_ptr.parm_name) then
                glob_pos_long_min (user_ptr);
            elsif s_equal
                     ("long_sec", user_ptr.parm_name) then
                glob_pos_long_sec (user_ptr);
            elsif s_equal
                      ("lat_deg", user_ptr.parm_name) then
                glob_pos_lat_deg (user_ptr);
            elsif s_equal
                       ("spacer", user_ptr.parm_name) then
                glob_pos_spacer (user_ptr);
            elsif s_equal
                     ("Spacer_1", user_ptr.parm_name) then
                glob_pos_Spacer_1 (user_ptr);
            elsif s_equal
                      ("lat_dir", user_ptr.parm_name) then
                glob_pos_lat_dir (user_ptr);
```

```
            elsif s_equal
                      ("long_dir", user_ptr.parm_name) then
                glob_pos_long_dir (user_ptr);

            end if;     -- END panel glob_pos

-- MODIFY TRACK PARAMETERS PANEL EVENTS

        elsif tae_wpt."=" (user_ptr, trk_mod_info)  then
            if s_equal
                  ("trk_identity", user_ptr.parm_name) then
                trk_mod_trk_identity (user_ptr);
            elsif s_equal
                   ("trk_control", user_ptr.parm_name) then
                trk_mod_trk_control (user_ptr);
            elsif s_equal
                ("trk_point_type", user_ptr.parm_name) then
                trk_mod_trk_point_type (user_ptr);
            elsif s_equal
                  ("trk_category", user_ptr.parm_name) then
                trk_mod_trk_category (user_ptr);
            elsif s_equal
                  ("trk_position", user_ptr.parm_name) then
                trk_mod_trk_position (user_ptr);
            elsif s_equal
                  ("trk_amp_info", user_ptr.parm_name) then
                trk_mod_trk_amp_info (user_ptr);
            elsif s_equal
                      ("trk_name", user_ptr.parm_name) then
                trk_mod_trk_name (user_ptr);
            elsif s_equal
```

```
                    ("trk_course", user_ptr.parm_name) then
                trk_mod_trk_course (user_ptr);
            elsif s_equal
                    ("trk_speed", user_ptr.parm_name) then
                trk_mod_trk_speed (user_ptr);
            elsif s_equal
                    ("trk_height", user_ptr.parm_name) then
                trk_mod_trk_height (user_ptr);
            elsif s_equal
                    ("trk_mod_enter", user_ptr.parm_name) then
                trk_mod_trk_mod_enter (user_ptr);
            elsif s_equal
                ("trk_plat_class", user_ptr.parm_name) then
                trk_mod_trk_plat_class (user_ptr);
        end if;     -- END panel trk_mod

-- CONFIRM DELETE PANEL EVENTS

        elsif tae_wpt."=" (user_ptr, confirm_info)  then
            if s_equal
                    ("conf_continue", user_ptr.parm_name) then
                confirm_conf_continue (user_ptr);
            elsif s_equal
                    ("conf_cancel", user_ptr.parm_name) then
                confirm_conf_cancel (user_ptr);
            end if;     -- END panel confirm

-- CONFIRM EXIT PANEL EVENTS

        elsif tae_wpt."=" (user_ptr, exit_ok_info)  then
            if s_equal
```

```
                ("exit_continue", user_ptr.parm_name) then
            abort Monitor_Update_Intervals;
            Integration_System.Shutdown;
            Write_Track_Archives_To_Text_File;
            Write_Filter_Archives_To_Text_File;
            abort Integration_System;
            Exit;
          --  exit_ok_exit_continue (user_ptr);
          elsif s_equal
                  ("exit_cancel", user_ptr.parm_name) then
            exit_ok_exit_cancel (user_ptr);
          end if;

-- INPUT ALERT PARAMETERS PANEL EVENTS

        elsif tae_wpt."=" (user_ptr, alrt_sel_info)  then
            if s_equal
                ("cpa_alrt_range", user_ptr.parm_name) then
                alrt_sel_cpa_alrt_range (user_ptr);
            elsif s_equal
                   ("alrt_canx_1", user_ptr.parm_name) then
                alrt_sel_alrt_canx_1 (user_ptr);
            elsif s_equal
                    ("enter_data", user_ptr.parm_name) then
                alrt_sel_enter_data (user_ptr);
            elsif s_equal
                   ("alrt_canx_2", user_ptr.parm_name) then
                alrt_sel_alrt_canx_2 (user_ptr);
            end if;

-- EQUIPMENT SELECTION PANEL EVENTS
```

```
elsif tae_wpt."=" (user_ptr, eqpt_sel_info) · then
    if s_equal
                ("gps_sel", user_ptr.parm_name) then
        eqpt_sel_gps_sel (user_ptr);
    elsif s_equal
              ("radar_sel", user_ptr.parm_name) then
        eqpt_sel_radar_sel (user_ptr);
    elsif s_equal
                ("link_sel", user_ptr.parm_name) then
        eqpt_sel_link_sel (user_ptr);
    elsif s_equal
                ("gyro_sel", user_ptr.parm_name) then
        eqpt_sel_gyro_sel (user_ptr);
    elsif s_equal
                ("fath_sel", user_ptr.parm_name) then
        eqpt_sel_fath_sel (user_ptr);
    elsif s_equal
                ("pit_sel", user_ptr.parm_name) then
        eqpt_sel_pit_sel (user_ptr);
    elsif s_equal
              ("enter_data", user_ptr.parm_name) then
        eqpt_sel_enter_data (user_ptr);
    end if;

-- TITLE PANEL EVENTS

elsif tae_wpt."=" (user_ptr, title_info)  then
  if s_equal
              ("exit_system", user_ptr.parm_name) then
        title_exit_system (user_ptr);
```

```
                end if;     -- END panel title

-- FILTER SELECT EVENTS

            elsif tae_wpt."=" (user_ptr, filt_sel_info)  then
                if s_equal
                        ("canx_filter", user_ptr.parm_name) then
                    filt_sel_canx_filter (user_ptr);
                elsif s_equal
                        ("type_select", user_ptr.parm_name) then
                    filt_sel_type_select (user_ptr);
                elsif s_equal
                   ("filter_complete", user_ptr.parm_name) then
                    filt_sel_filter_complete (user_ptr);
                elsif s_equal
                        ("add_filter", user_ptr.parm_name) then
                    filt_sel_add_filter (user_ptr);
                end if;     -- END panel filt_sel

-- RANGE FILTER EVENTS

            elsif tae_wpt."=" (user_ptr, rng_filt_info)  then
                if s_equal
                          ("rng_type", user_ptr.parm_name) then
                    rng_filt_rng_type (user_ptr);
                elsif s_equal
                          ("rng_limit", user_ptr.parm_name) then
                    rng_filt_rng_limit (user_ptr);
                elsif s_equal
                          ("hgt_limit", user_ptr.parm_name) then
                    rng_filt_hgt_limit (user_ptr);
```

```
                elsif s_equal
                        ("operator", user_ptr.parm_name) then
                    rng_filt_operator (user_ptr);
                elsif s_equal
                        ("complete", user_ptr.parm_name) then
                    rng_filt_complete (user_ptr);
                elsif s_equal
                        ("cancel", user_ptr.parm_name) then
                    rng_filt_cancel (user_ptr);
                end if;     -- END panel rng_filt

-- CATEGORY FILTER EVENTS

            elsif tae_wpt."=" (user_ptr, cat_filt_info)  then
                if s_equal
                        ("cancel", user_ptr.parm_name) then
                    cat_filt_cancel (user_ptr);
                elsif s_equal
                        ("cat_type", user_ptr.parm_name) then
                    cat_filt_cat_type (user_ptr);
                elsif s_equal
                        ("operator", user_ptr.parm_name) then
                    cat_filt_operator (user_ptr);
                end if;     -- END panel cat_filt

-- IDENTITY FILTER EVENTS

            elsif tae_wpt."=" (user_ptr, id_filt_info)  then
                if s_equal
                        ("id_type", user_ptr.parm_name) then
                    id_filt_id_type (user_ptr);
```

```
                elsif s_equal
                            ("cancel", user_ptr.parm_name) then
                    id_filt_cancel (user_ptr);
                elsif s_equal
                          ("Operator", user_ptr.parm_name) then
                    id_filt_Operator (user_ptr);
                end if;     -- END panel id_filt

-- INTERCEPT EVENTS

            elsif tae_wpt."=" (user_ptr, int_cept_info)  then
                if s_equal
                            ("method", user_ptr.parm_name) then
                    int_cept_method (user_ptr);
                elsif s_equal
                             ("time", user_ptr.parm_name) then
                    int_cept_time (user_ptr);
                elsif s_equal
                             ("speed", user_ptr.parm_name) then
                    int_cept_speed (user_ptr);
                elsif s_equal
                           ("compute", user_ptr.parm_name) then
                    int_cept_compute (user_ptr);
                elsif s_equal
                            ("cancel", user_ptr.parm_name) then
                    int_cept_cancel (user_ptr);
                end if;     -- END panel int_cept
             else
              null;
            -- or raise an exception, but compiler warns if no exit
             end if;
```

```
-- TAE FILE EVENTS STUB
            when tae_wpt.WPT_FILE_EVENT =>
              text_io.put_line ("STUB: Event WPT_FILE_EVENT");
-- TAE TIMEOUT EVENT STUB
            when tae_wpt.WPT_TIMEOUT_EVENT =>
              text_io.put_line ("STUB: Event WPT_TIMEOUT_EVENT");
-- MISC TAE EVENTS
            when tae_wpt.WPT_WINDOW_EVENT => null ;
            when tae_wpt.WPT_HELP_EVENT =>   null ;
            when tae_wpt.WPT_INTERRUPT_EVENT => null ;
            when OTHERS => null;
        end case;
      end if;
   end loop;
end menu;
```

APPENDIX G

LCCDS USERS MANUAL

A. STARTING THE SYSTEM

To invoke the system from the command line simply type lccds.out at any command prompt. X11R4 and a window manager (such as Open Look or TWM) must be running prior to calling LCCDS.

The program will start by opening all the display windows, the Menu bar and the System Information box. The display windows can then be moved to any location desired by the user by "dragging" the display box with the mouse. The Menu bar and the System Information box are static and cannot be moved.

The prototype model pre-loads an ownship track and 3 target test tracks for demonstration purposes. These test tracks can be modified or deleted as the user desires. The ownship track may be modified, but cannot be deleted.

B. OPERATING THE SYSTEM

1. Tactical Plot Display Functions

The Tactical Plot display window, shown in Figure G-1, contains a number of information items, and a graphical representation of all tracks currently in the system.

Each track in the system will be displayed as a standard GOTS symbol with a "speed leader", or line, emanating from the center of the symbol denoting true course and speed. The system track number will be shown in a box directly below the symbol.

Most user interaction with the system will occur in this window. There are 2 interactive functions built into this window:

- Track Selection
- Display Range Scaling

a. Track Selection

To select an existing track, move the mouse cursor over the graphical object and press any mouse button. The display will immediately update and a "hook", or circle, will appear surrounding the object. All information currently available in the system concerning the hooked object will appear in the Track Information and Intelligence displays, shown in Figures G-2 and G-3 respectively.

b. Range Scaling

The upper left corner of the Tactical Plot display contains a button pair labelled "DOWN" and "UP". Moving the mouse cursor to either of these buttons, and pressing any mouse button will immediately update the Tactical Plot display and either halve or double the range representation of the screen, depending on the button selected. The Scale information item above the buttons will reflect the current range scale of the display screen.

2. Menu Bar Functions

In the current configuration there are 5 menu pulldown buttons with active functions:

- Tracks Button
- Alerts Button
- Defaults Button
- Filters Button
- Nav Button

Selection of an inactive menu function will result in the display of a dialog box message in the upper right corner of the display, informing the user that the item selected was included for future development.

a. Track Functions

The options available in this pulldown option include adding a Track, deleting a hooked Track and modifying a hooked Track. Selection of the latter 2 options without a track actively hooked will result in no action.

Selection of the Add a Track option results in a dialog box, shown in Figure G-4, opening asking the user for the add method desired. The methods available are add by bearing and range or add by global position. Additional dialog boxes, shown in Figures G-5 and G-6 will appear to accept the user input for those methods.

Selection of the Delete the Hooked Track option will, if a track is currently hooked, display a dialog box asking the user to confirm the deletion. If the "Continue" box is selected, the Track will be deleted on the next screen update. The "Cancel" option will dismiss the dialog box with no action taken.

Selection of the Modify the Hooked Track option will result in the dialog box, shown in Figure G-7, opening to allow the user to input the attribute changes desired. Any changes made will take effect immediately and will be displayed on the next screen update. The "Enter Data" button will dismiss the dialog box and return the user to the normal display.

b. Alerts Functions

The 3 pulldown options, Visual Alerts Only, Audio and Visual Alerts and Disable all Alerts will take the appropriate action immediately upon selection. The Alerts information item in the upper right corner of the Tactical Plot will reflect the current status of alerts. The default configuration is both audio and visual alerts enabled, with any alert conditions detected by the system shown in the Alerts display box in Figure G-8, accompanied by a "beep" at every alert detection.

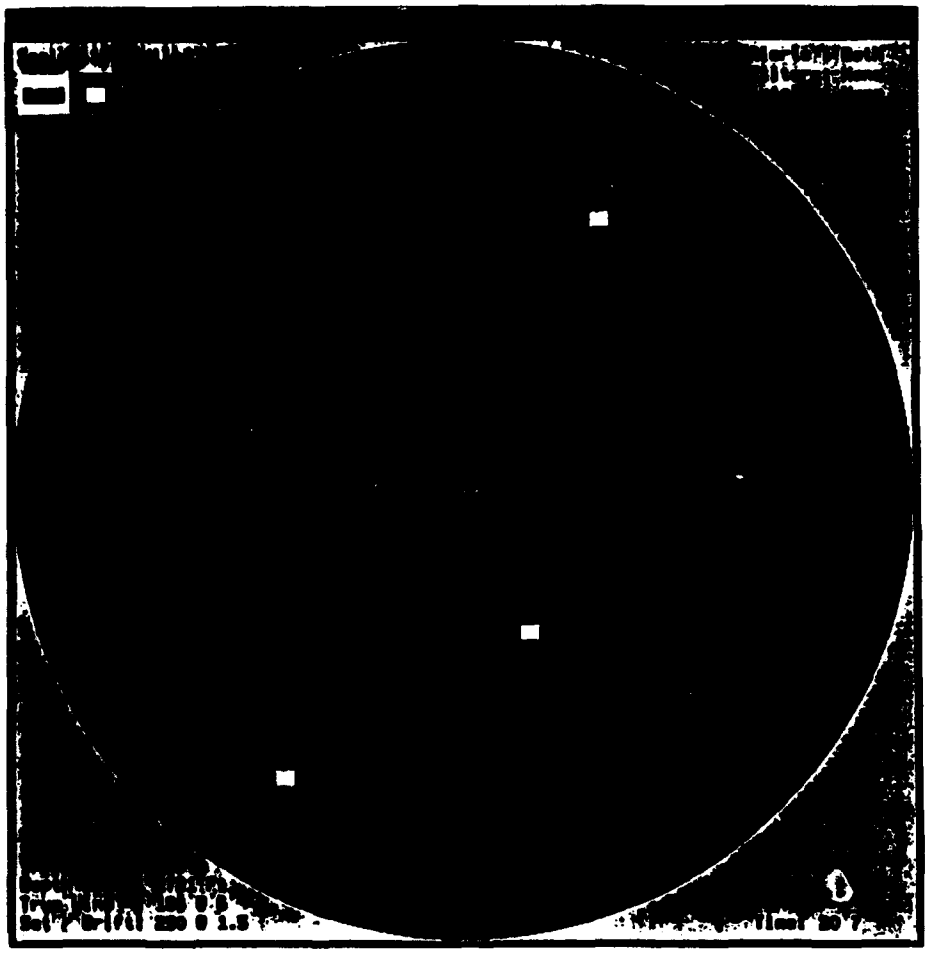

Figure G-1. Tactical Plot Display Window

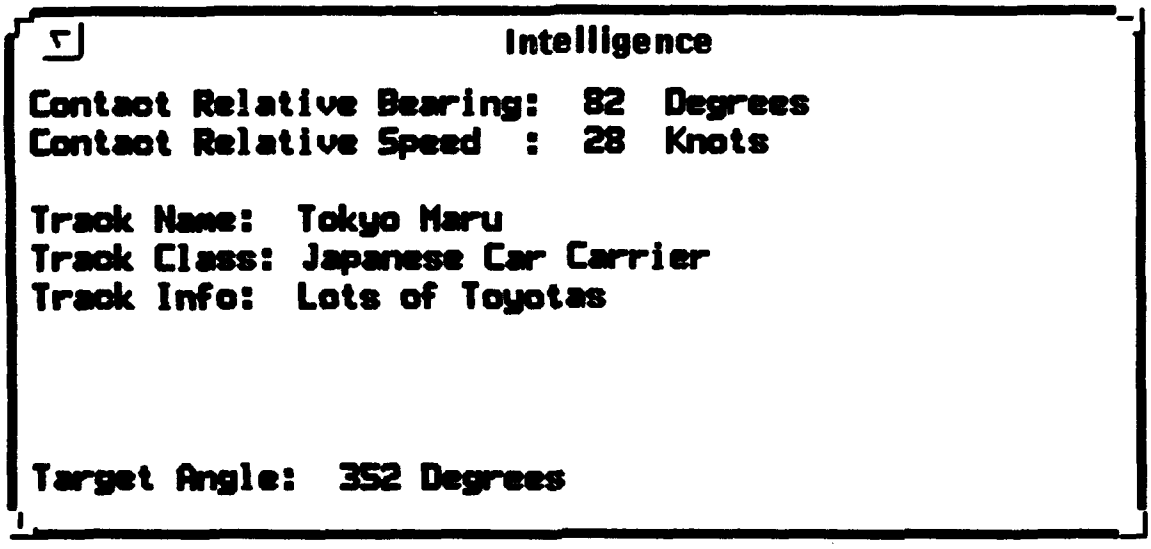

Figure G-2. Intelligence Display Window

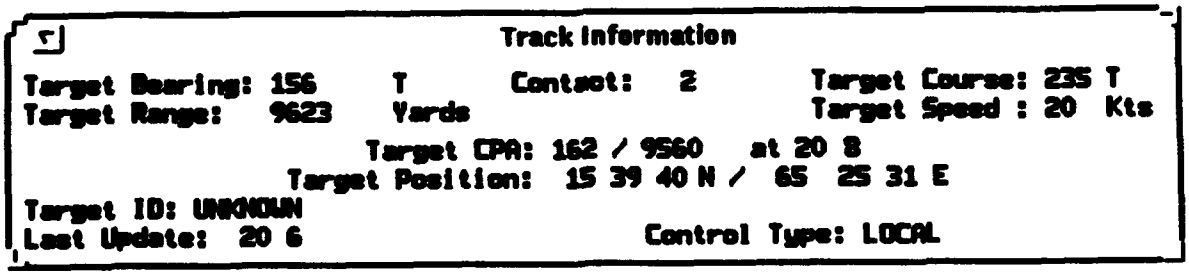

Figure G-3. Track Information Display Window

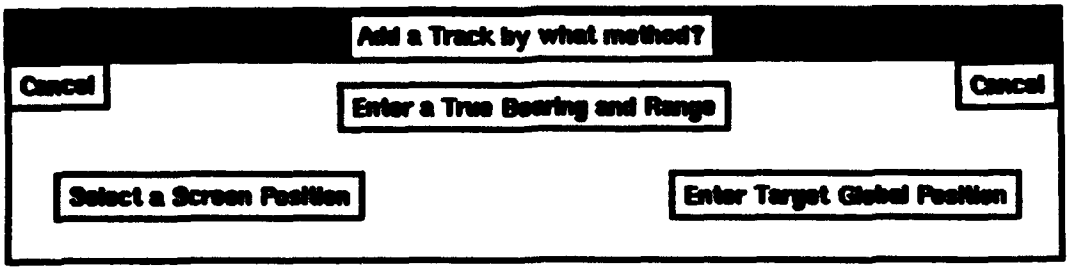

Figure G-4. Add a Track Dialog Box

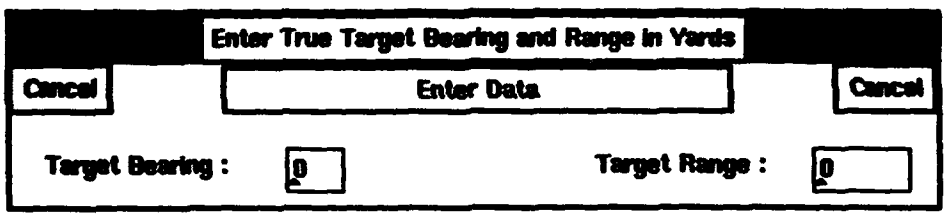

Figure G-5. Enter Bearing and Range Data Entry Box

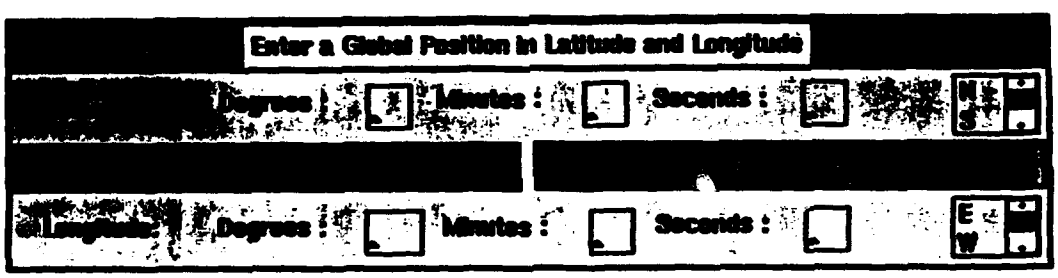

Figure G-6. Enter Global Position Data Entry Box

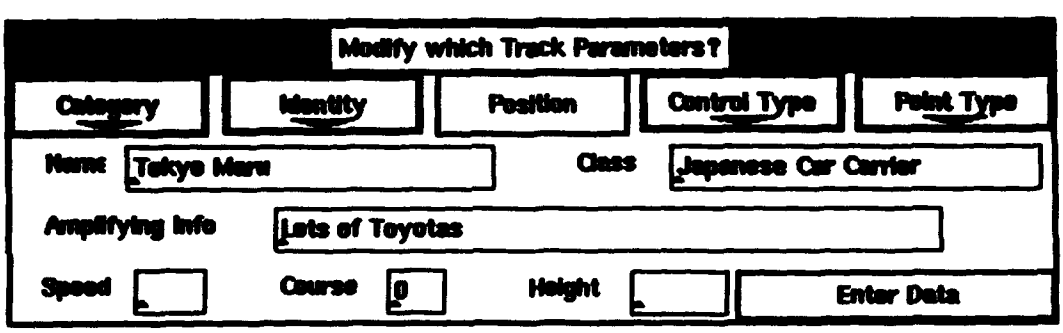

Figure G-7. Modify Track Parameters Data Entry Box

c. Filters Functions

Selection of either option within this pulldown will immediately discard any display filter conditions set by the user.

d. Defaults Functions

(1) Set Display Filters. Selection of the Set Display Filters option within this pulldown will present the user with the dialog box shown in Figure G-8.

The set filter operation is a recursive routine allowing the user to define multiple atomic filters within a single composite filter object. An atomic filter is defined on the attributes of range, height, category or identity. A series of data input boxes, shown in Figures G-9 through G-11 allow the user to define a single atomic filter, always returning the user to the dialog box at the completion of each data entry item.

Selection of the Add Filter Element button in the dialog box will add the atomic filter to the composite filter object and reset the atomic entry routine, allowing the user to define another atomic filter.

Selection of the Atomic Filter Completed button will close the filter, and return the user to the main display. Any further filter additions can be made by reinitializing the process.

All system filters can be cleared immediately by selection of either option within the Filters menu pulldown.

(2) Set Alert Parameters. Selection of the Set Alert Parameters pulldown item will present the user with the data entry box shown in Figure G-12. Input of a value

into the entry box and the selection of Enter Data will immediately apply the new CPA alert value into the system. Any Track with a CPA range less than the alert range will generate an Alert in the Alerts display window shown in Figure G-13, provided the Alerts are not disabled by an Alerts menu button action.

(3) Set External System Inputs. Selection of this option will present the user with the checkbox option window shown in Figure G-14. Selection of an item will place an X into the checkbox, signifying the activation of that piece of equipment's inputs into the system. Conversely, the deselecting of a checked box will remove the X and signify the deactivation of that equipment's inputs into the system.

The Enter Data button will dismiss the window and make the necessary system modifications. The System Status display window, shown in Figure G-15, will reflect any changes made to the system inputs on the next display update.

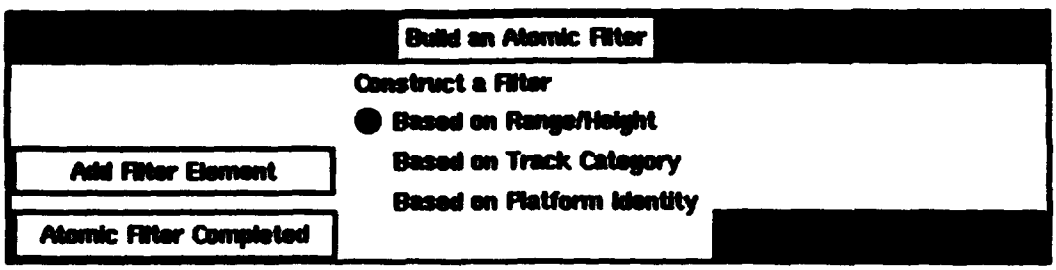

Figure G-8. Build an Atomic Filter Dialog Box

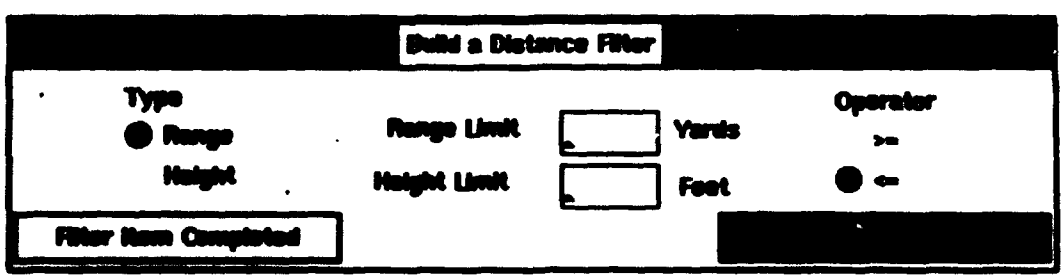

Figure G-9. Build a Distance Filter

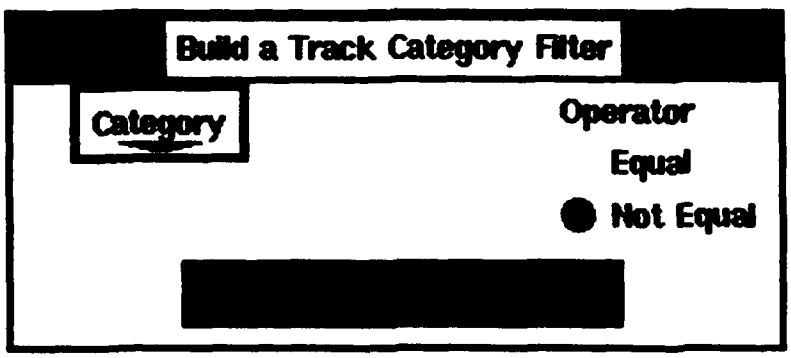

Figure G-10. Build a Track Category Filter

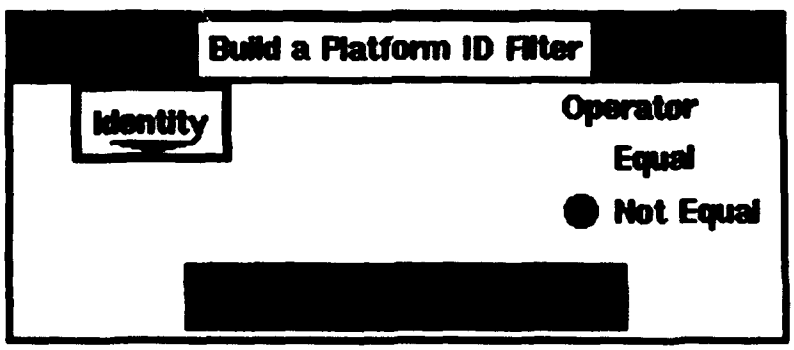

Figure G-11. Build a Platform ID Filter

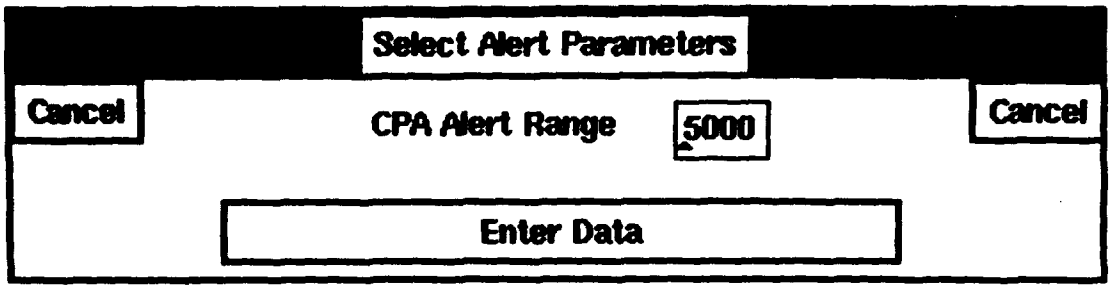

Figure G-12. Select Alert Parameters

Figure G-13. Alerts Display Window

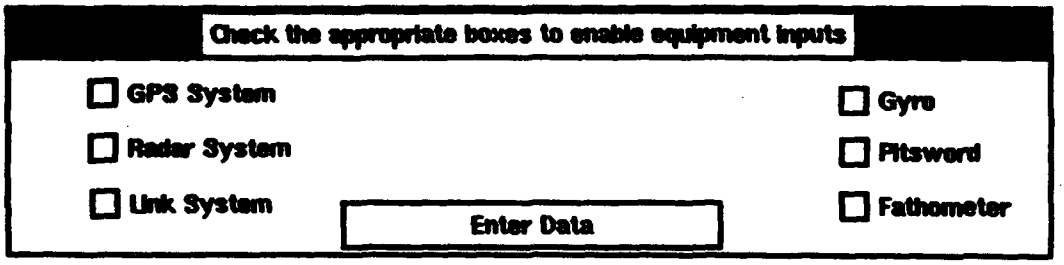

Figure G-14. Equipment Selection Options

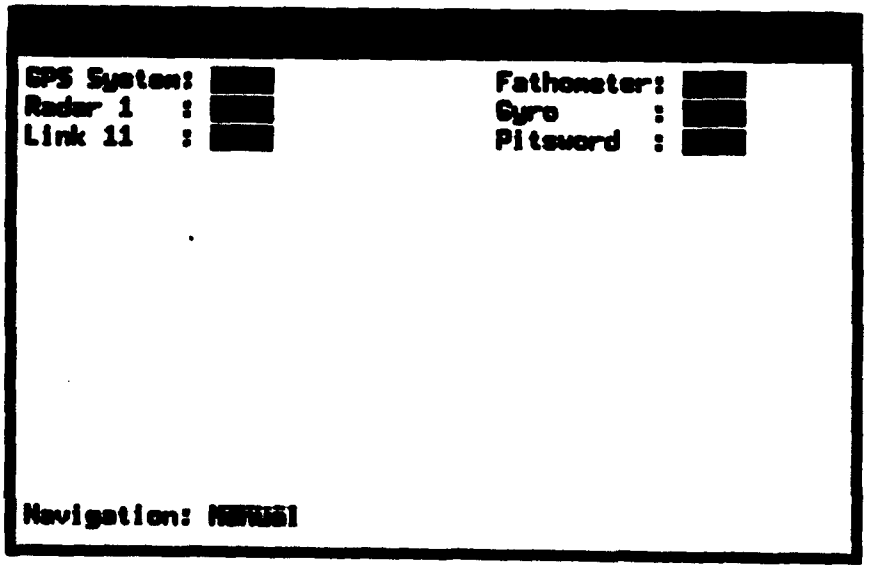

Figure G-15. System Status Display Window

e. Nav Functions

The only active function within this menu pulldown is Plot an Intercept. A Track must be hooked prior to activating this function or no action will occur. Selection of this option will present the user with a data entry box, shown in Figure G-16, allowing the user to compute an Intercept to the hooked track by 2 different methods.

(1) Time to Station Method. This, the default method, will compute an intercept to the hooked track based on time. Entering of the number of minutes to intercept by the user, and the selection of the Compute button, will return the course and speed necessary to effect the intercept in the desired interval. All intercept recommendations will appear in the Recommendations display window shown in Figure G-17.

(2) Speed to Station Method. This method will take a projected speed input from the user and return the course necessary for the intercept and the estimated time the intercept will take place.

The Compute button will initiate all computations. The Cancel button will return the user to the default display. Any active recommendation within the Recommendations Display Window can be dismissed by moving the cursor to the window and clicking any mouse button.

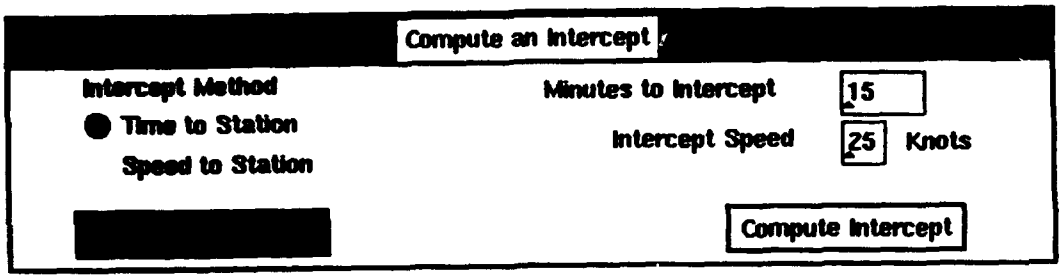

Figure G-16. Compute an Intercept

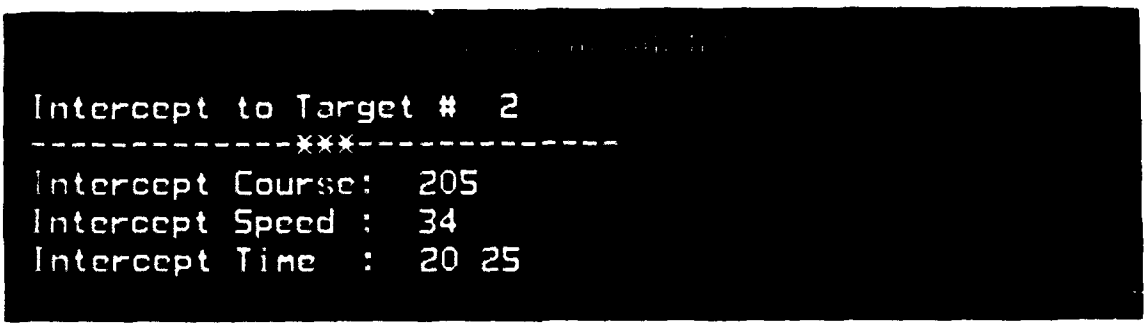

Figure G-17. Recommendations Display Window

C. EXITING THE SYSTEM

An Exit System button is provided on the System Information window shown in Figure G-18. Selection of the button will present the user with the confirmation dialog box shown in Figure G-19. Selection of the Continue option within the dialog box will shut the system down and write all Track Histories, Filter Histories and Observations to text files within the current sub-directory. Selection of the Cancel option will result in dismissal of the dialog box and no system action taking place.

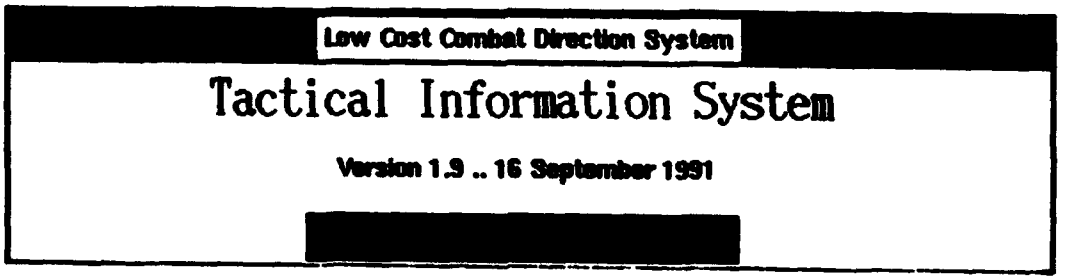

Figure G-18. System Information Window

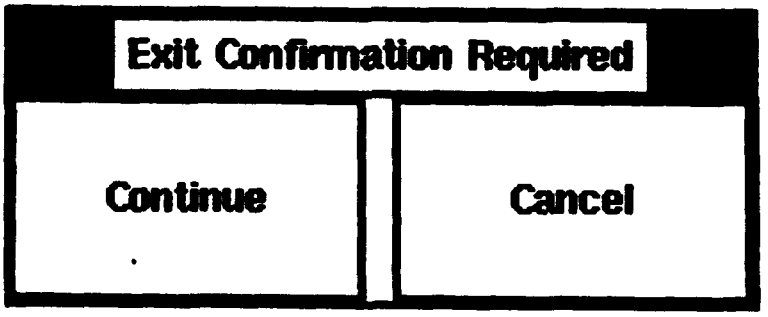

Figure G-19. Exit Confirmation Dialog Box

LIST OF REFERENCES

[ADAXT] Unisys Defense Systems Inc., Reston, VA; ***Version Description Document for the Software Technology for Adaptable, Reliable Systems (STARS)***; Ada/Xt Toolkit, Version 3.1, Sun OS, STARS-SC-03716/001/00, Publication No. GR-7670-1239(NP), May 1991.

[BI91] Bolick, W. & Irwin, R; ***The Tactical Database for the Low Cost Combat Direction System***; Masters Thesis, Naval Postgraduate School, Monterey, CA, September 1991.

[CK90] Coskun, V. & Kesoglu, C.; ***A Software Prototype for a Command, Control, Communications and Intelligence (C3I) Workstation***; Masters Thesis, Naval Postgraduate School, Monterey, CA, December 1990.

[GOTS91] Commander, Space and Naval Warfare Systems Command; ***GOTS Version 1.0.5 Technical Manual, (series);*** January 1991.

[SOW88] Commander, Naval Sea Systems Command; UNCLASSIFIED letter 9410, OPR:61Y, Serial 61Y1036 to Superintendent, Naval Postgraduate School, Monterey, CA. Subject: ***Statement of Work for Low Cost Combat Direction System***, 20 December 1988.

[SS90] Seveney, J. & Steinberg, G.; ***Requirements Analysis for a Low Cost Combat Direction System***; Masters Thesis, Naval Postgraduate School, Monterey, CA, June 1990.

[SUN90] Sun, C.; ***Developing Portable User Interfaces for Ada Command Control Software***; Masters Thesis, Naval Postgraduate School, Monterey, CA, June 1990.

[TAE] National Aeronautics and Space Administration, Goddard Space Flight Center; ***Transportable Applications Environment (TAE) Plus Technical Manual, (series);*** January 1990.

[XLIB] Unisys Defense Systems Inc., Reston, VA; ***Version Description Document for the Software Technology for Adaptable, Reliable Systems (STARS)***; Ada/Xlib Bindings, Version 1.0, Sun OS, STARS-SS-022007/001/00, Publication No. GR-7670-1181(NP), January 1991.

INITIAL DISTRIBUTION LIST

Defense Technical Information Center Cameron Station Alexandria, VA 22304	2
Dudley Knox Library Code 1424 Naval Postgraduate School Monterey, CA 93943	2
Director of Research Administration Code 012 Naval Postgraduate School Monterey, CA 93943	1
Office of Naval Research 800 N. Quincy Street Arlington, VA 22217	1
Office of the Chief of Naval Operations Code OP-941 Washington, DC 20350	1
Commander Naval Sea Systems Command ATTN: LCDR Scott Kelly Code 06D3131 Washington, DC 20362	1
Assistant Secretary of the Navy Research, Development and Acquisitions Washington, DC 20350	1
Commander Naval Ocean Systems Center Code 451 San Diego, CA 92152	1

Commander
Naval Ocean Systems Center
Code 431
ATTN: Dan Edwards
San Diego, CA 92152 — 1

Commander
Naval Sea Systems Command
ATTN: William L. Wilder
PMS-4123H
Arlington, VA 22202 — 1

Dr. Valdis Berzins
Code 52Be
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943 — 1

Dr. Luqi
Code 52Lq
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943 — 1

LT Michael G. Stockwell
121 Rome Drive
Vallejo, CA 94589 — 1